

Recitation 2: PM/0 Code Execution

COP3402 FALL 2015 – ARYA POURTABATABAIE
FROM EURIPIDES MONTAGNE, FALL 2014

Review and Reference: The PM/O Instruction Set

Op	Mnemonic	Description
01	LIT 0, M	Push the literal value M onto the stack.
02	OPR 0, 0	Return from a procedure call.
02	OPR 0, M	Perform an ALU operation, specified by M.
03	LOD L, M	Read the value at offset M from L levels down (if L=0, our own frame) and push it onto the stack.
04	STO L, M	Pop the stack and write the value into offset M from L levels down – if L=0, our own frame.
05	CAL L, M	Call the procedure at M.
06	INC 0, M	Allocate enough space for M local variables. We will always allocate at least four.
07	JMP 0, M	Branch to M.
08	JPC 0, M	Pop the stack and branch to M if the result is 0.
09	SIO 0, 1	Pop the stack and write the result to the screen.
10	SIO 0, 2	Read an input from the user and store it at the top of the stack.
11	SIO 0, 3	Stop the machine.

Review and Reference: ALU Operations

Operation	Name	Description
OPR 0, 1	NEG	Pop the stack and push the negation of the result.
OPR 0, 2	ADD	Pop the stack twice, add the values, and push the result.
OPR 0, 3	SUB	Pop the stack twice, subtract the top value from the second value, and push the result.
OPR 0, 4	MUL	Pop the stack twice, multiply the values, and push the result.
OPR 0, 5	DIV	Pop the stack twice, divide the second value by the top value, and push the quotient.
OPR 0, 6	ODD	Pop the stack, push 1 if the value is odd, and push 0 otherwise.
OPR 0, 7	MOD	Pop the stack twice, divide the second value by the top value, and push the remainder.
OPR 0, 8	EQL	Pop the stack twice and compare the top value t with the second value s . Push 1 if $s = t$ and 0 otherwise.
OPR 0, 9	NEQ	Pop the stack twice and compare the top value t with the second value s . Push 1 if $s \neq t$ and 0 otherwise.
OPR 0, 10	LSS	Pop the stack twice and compare the top value t with the second value s . Push 1 if $s < t$ and 0 otherwise.
OPR 0, 11	LEQ	Pop the stack twice and compare the top value t with the second value s . Push 1 if $s \leq t$ and 0 otherwise.
OPR 0, 12	GTR	Pop the stack twice and compare the top value t with the second value s . Push 1 if $s > t$ and 0 otherwise.
OPR 0, 13	GEQ	Pop the stack twice and compare the top value t with the second value s . Push 1 if $s \geq t$ and 0 otherwise.

Review and Reference: Instruction Pseudocode

Op	Mnemonic	Pseudocode
01	LIT 0, M	$sp \leftarrow sp + 1;$ $stack[sp] \leftarrow M;$
02	OPR 0, 0 (Return)	$sp \leftarrow bp - 1;$ $pc \leftarrow stack[sp + 4];$ $bp \leftarrow stack[sp + 3];$
03	LOD L, M	$sp \leftarrow sp + 1;$ $stack[sp] \leftarrow stack[base(L) + M];$
04	STO L, M	$stack[base(L) + M] \leftarrow stack[sp];$ $sp \leftarrow sp - 1;$
05	CAL L, M	<i>/* FV, SL, DL, RA */</i> $stack[sp + 1] \leftarrow 0;$ $stack[sp + 2] \leftarrow base(L);$ $stack[sp + 3] \leftarrow bp;$ $stack[sp + 4] \leftarrow pc;$ $bp \leftarrow sp + 1;$ $pc \leftarrow M;$

Op	Mnemonic	Pseudocode
06	INC 0, M	$sp \leftarrow sp + M;$
07	JMP 0, M	$pc = M;$
08	JPC 0, M	if $stack[sp] == 0$ then { $pc \leftarrow M;$ } $sp \leftarrow sp - 1;$
09	SIO 0, 1	print ($stack[sp]$); $sp \leftarrow sp - 1;$
10	SIO 0, 2	$sp \leftarrow sp + 1;$ read ($stack[sp]$);
11	SIO 0, 3	halt;

Base(L) is the base of the stack frame L levels down from ours.
If L is 0, it's our own frame.

Review and Reference: The Compiled Example

	Line	OP	L	M
const n = 13; /* constant declaration	0	jmp	0	10
var i, h; /* variable declaration	1	jmp	0	2
procedure sub;				
const k = 7;	2	inc	0	6
var j, h;	3	lit	0	13
begin	4	sto	0	4
j := n;	5	lit	0	1
i := 1;	6	sto	1	4
h := k;	7	lit	0	7
end;	8	sto	0	5
	9	opr	0	0
begin /* main starts here				
i := 3;	10	inc	0	6
h := 0;	11	lit	0	3
call sub;	12	sto	0	4
end.	13	lit	0	0
	14	sto	0	5
	15	cal	0	2
	16	sio	0	3

Review and Reference: Running a Program...

Initial values				pc	bp	sp	stack
				0	1	0	
0	jmp	0	10	10	1	0	
10	inc	0	6	11	1	6	0 0 0 0 0 0
11	lit	0	3	12	1	7	0 0 0 0 0 0 3
12	sto	0	4	13	1	6	0 0 0 0 3 0
13	lit	0	0	14	1	7	0 0 0 0 3 0 0
14	sto	0	5	15	1	6	0 0 0 0 3 0
15	cal	0	2	2	7	6	0 0 0 0 3 0
2	inc	0	6	3	7	12	0 0 0 0 3 0 0 1 1 16 0 0
3	lit	0	13	4	7	13	0 0 0 0 3 0 0 1 1 16 0 0 13
4	sto	0	4	5	7	12	0 0 0 0 3 0 0 1 1 16 13 0
5	lit	0	1	6	7	13	0 0 0 0 3 0 0 1 1 16 13 0 1
6	sto	1	4	7	7	12	0 0 0 0 1 0 0 1 1 16 13 0
7	lit	0	7	8	7	13	0 0 0 0 1 0 0 1 1 16 13 0 7
8	sto	0	5	9	7	12	0 0 0 0 1 0 0 1 1 16 13 7
9	opr	0	0	16	1	6	0 0 0 0 1 0
16	sio	0	3	0	0	0	

Review and Reference: Running a Program...

Initial values		pc		bp		sp		stack	
		0		1		0			
0	jmp	0	10	10	1	0			
10	inc	0	6	11	1	6	0 0 0 0 0 0		
11	lit								
12	sto								
13	lit								
14	sto								
15	cal								
2	inc								0 0
3	lit								0 0 13
4	sto								13 0
5	lit								13 0 1
6	sto								13 0
7	lit								13 0 7
8	sto	0	5	9	7	12	0 0 0 0 1 0	0 1 1 16	13 7
9	opr	0	0	16	1	6	0 0 0 0 1 0		
16	sio	0	3	0	0	0			

...but how does that happen?

*...but how does
that happen?*

Running Code on PM/0

PM/0 State: Begin

Reg	Value	Stack	Value
IR		1	0
PC	0	2	0
BP	1	3	0
SP	0	4	0
		5	0
		6	0
		7	0
		8	0
		9	0
		10	0
		11	0
		12	0
		13	0
		14	0
		15	0

Line	OP	L	M
0	jmp	0	10
1	jmp	0	2
2	inc	0	6
3	lit	0	13
4	sto	0	4
5	lit	0	1
6	sto	1	4
7	lit	0	7
8	sto	0	5
9	opr	0	0
10	inc	0	6
11	lit	0	3
12	sto	0	4
13	lit	0	0
14	sto	0	5
15	cal	0	2
16	sio	0	3

PM/0 State: Fetch

Reg	Value	Stack	Value
IR	JMP 0 10	1	0
PC	1	2	0
BP	1	3	0
SP	0	4	0
		5	0
		6	0
		7	0
		8	0
		9	0
		10	0
		11	0
		12	0
		13	0
		14	0
		15	0

Line	OP	L	M
0	jmp	0	10
1	jmp	0	2
2	inc	0	6
3	lit	0	13
4	sto	0	4
5	lit	0	1
6	sto	1	4
7	lit	0	7
8	sto	0	5
9	opr	0	0
10	inc	0	6
11	lit	0	3
12	sto	0	4
13	lit	0	0
14	sto	0	5
15	cal	0	2
16	sio	0	3

PM/0 State: Execute JMP

pc = M;

Reg	Value	Stack	Value
IR	JMP 0 10	1	0
PC	10	2	0
BP	1	3	0
SP	0	4	0
		5	0
		6	0
		7	0
		8	0
		9	0
		10	0
		11	0
		12	0
		13	0
		14	0
		15	0

Line	OP	L	M
0	jmp	0	10
1	jmp	0	2
2	inc	0	6
3	lit	0	13
4	sto	0	4
5	lit	0	1
6	sto	1	4
7	lit	0	7
8	sto	0	5
9	opr	0	0
10	inc	0	6
11	lit	0	3
12	sto	0	4
13	lit	0	0
14	sto	0	5
15	cal	0	2
16	sio	0	3

PM/0 State: Fetch

Reg	Value	Stack	Value
IR	INC 0 6	1	0
PC	11	2	0
BP	1	3	0
SP	0	4	0
		5	0
		6	0
		7	0
		8	0
		9	0
		10	0
		11	0
		12	0
		13	0
		14	0
		15	0

Line	OP	L	M
0	jmp	0	10
1	jmp	0	2
2	inc	0	6
3	lit	0	13
4	sto	0	4
5	lit	0	1
6	sto	1	4
7	lit	0	7
8	sto	0	5
9	opr	0	0
10	inc	0	6
11	lit	0	3
12	sto	0	4
13	lit	0	0
14	sto	0	5
15	cal	0	2
16	sio	0	3

PM/0 State: Execute INC

$sp \leftarrow sp + M;$

Reg	Value	Stack	Value
IR	INC 0 6	1	0
PC	11	2	0
BP	1	3	0
SP	6	4	0
		5	0
		6	0
		7	0
		8	0
		9	0
		10	0
		11	0
		12	0
		13	0
		14	0
		15	0

Line	OP	L	M
0	jmp	0	10
1	jmp	0	2
2	inc	0	6
3	lit	0	13
4	sto	0	4
5	lit	0	1
6	sto	1	4
7	lit	0	7
8	sto	0	5
9	opr	0	0
10	inc	0	6
11	lit	0	3
12	sto	0	4
13	lit	0	0
14	sto	0	5
15	cal	0	2
16	sio	0	3

PM/0 State: Fetch

Reg	Value	Stack	Value
IR	LIT 0 3	1	0
PC	12	2	0
BP	1	3	0
SP	6	4	0
		5	0
		6	0
		7	0
		8	0
		9	0
		10	0
		11	0
		12	0
		13	0
		14	0
		15	0

Line	OP	L	M
0	jmp	0	10
1	jmp	0	2
2	inc	0	6
3	lit	0	13
4	sto	0	4
5	lit	0	1
6	sto	1	4
7	lit	0	7
8	sto	0	5
9	opr	0	0
10	inc	0	6
11	lit	0	3
12	sto	0	4
13	lit	0	0
14	sto	0	5
15	cal	0	2
16	sio	0	3

PM/0 State: Execute LIT

$sp \leftarrow sp + 1;$
 $stack[sp] \leftarrow M;$

Reg	Value	Stack	Value
IR	LIT 0 3	1	0
PC	12	2	0
BP	1	3	0
SP	7	4	0
		5	0
		6	0
		7	3
		8	0
		9	0
		10	0
		11	0
		12	0
		13	0
		14	0
		15	0

Line	OP	L	M
0	jmp	0	10
1	jmp	0	2
2	inc	0	6
3	lit	0	13
4	sto	0	4
5	lit	0	1
6	sto	1	4
7	lit	0	7
8	sto	0	5
9	opr	0	0
10	inc	0	6
11	lit	0	3
12	sto	0	4
13	lit	0	0
14	sto	0	5
15	cal	0	2
16	sio	0	3

PM/0 State: Execute STO

$\text{stack}[\text{base}(\text{L})+\text{M}] \leftarrow \text{stack}[\text{sp}];$
 $\text{sp} \leftarrow \text{sp} - 1;$

Reg	Value	Stack	Value
IR	STO 0 4	1	0
PC	13	2	0
BP	1	3	0
SP	6	4	0
		5	3
		6	0
		7	0
		8	0
		9	0
		10	0
		11	0
		12	0
		13	0
		14	0
		15	0

Line	OP	L	M
0	jmp	0	10
1	jmp	0	2
2	inc	0	6
3	lit	0	13
4	sto	0	4
5	lit	0	1
6	sto	1	4
7	lit	0	7
8	sto	0	5
9	opr	0	0
10	inc	0	6
11	lit	0	3
12	sto	0	4
13	lit	0	0
14	sto	0	5
15	cal	0	2
16	sio	0	3

PM/0 State: Execute LIT

$sp \leftarrow sp + 1;$
 $stack[sp] \leftarrow M;$

Reg	Value	Stack	Value
IR	LIT 0 0	1	0
PC	14	2	0
BP	1	3	0
SP	7	4	0
		5	3
		6	0
		7	0
		8	0
		9	0
		10	0
		11	0
		12	0
		13	0
		14	0
		15	0

Line	OP	L	M
0	jmp	0	10
1	jmp	0	2
2	inc	0	6
3	lit	0	13
4	sto	0	4
5	lit	0	1
6	sto	1	4
7	lit	0	7
8	sto	0	5
9	opr	0	0
10	inc	0	6
11	lit	0	3
12	sto	0	4
13	lit	0	0
14	sto	0	5
15	cal	0	2
16	sio	0	3

PM/0 State: Execute STO

$\text{stack}[\text{base}(\text{L})+\text{M}] \leftarrow \text{stack}[\text{sp}];$
 $\text{sp} \leftarrow \text{sp} - 1;$

Reg	Value	Stack	Value
IR	STO 0 5	1	0
PC	15	2	0
BP	1	3	0
SP	6	4	0
		5	3
		6	0
		7	0
		8	0
		9	0
		10	0
		11	0
		12	0
		13	0
		14	0
		15	0

Line	OP	L	M
0	jmp	0	10
1	jmp	0	2
2	inc	0	6
3	lit	0	13
4	sto	0	4
5	lit	0	1
6	sto	1	4
7	lit	0	7
8	sto	0	5
9	opr	0	0
10	inc	0	6
11	lit	0	3
12	sto	0	4
13	lit	0	0
14	sto	0	5
15	cal	0	2
16	sio	0	3

PM/0 State: Fetch

Reg	Value	Stack	Value
IR	CAL 0 2	1	0
PC	16	2	0
BP	1	3	0
SP	6	4	0
		5	3
		6	0
		7	0
		8	0
		9	0
		10	0
		11	0
		12	0
		13	0
		14	0
		15	0

Line	OP	L	M
0	jmp	0	10
1	jmp	0	2
2	inc	0	6
3	lit	0	13
4	sto	0	4
5	lit	0	1
6	sto	1	4
7	lit	0	7
8	sto	0	5
9	opr	0	0
10	inc	0	6
11	lit	0	3
12	sto	0	4
13	lit	0	0
14	sto	0	5
15	cal	0	2
16	sio	0	3

PM/0 State: Execute CAL

```
/* FV, SL, DL, RA */
stack[sp+1] ← 0;
stack[sp+2] ← base(L);
stack[sp+3] ← bp;
stack[sp+4] ← pc;
bp ← sp + 1;
pc ← M;
```

Reg	Value	Stack	Value
IR	CAL 0 2	1	0
PC	2	2	0
BP	7	3	0
SP	6	4	0
		5	3
		6	0
		7	0
		8	1
		9	1
		10	16
		11	0
		12	0
		13	0
		14	0
		15	0

Line	OP	L	M
0	jmp	0	10
1	jmp	0	2
2	inc	0	6
3	lit	0	13
4	sto	0	4
5	lit	0	1
6	sto	1	4
7	lit	0	7
8	sto	0	5
9	opr	0	0
10	inc	0	6
11	lit	0	3
12	sto	0	4
13	lit	0	0
14	sto	0	5
15	cal	0	2
16	sio	0	3

PM/0 State: Execute INC

$sp \leftarrow sp + M;$

Reg	Value	Stack	Value
IR	INC 0 6	1	0
PC	3	2	0
BP	7	3	0
SP	12	4	0
		5	3
		6	0
		7	0
		8	1
		9	1
		10	16
		11	0
		12	0
		13	0
		14	0
		15	0

Line	OP	L	M
0	jmp	0	10
1	jmp	0	2
2	inc	0	6
3	lit	0	13
4	sto	0	4
5	lit	0	1
6	sto	1	4
7	lit	0	7
8	sto	0	5
9	opr	0	0
10	inc	0	6
11	lit	0	3
12	sto	0	4
13	lit	0	0
14	sto	0	5
15	cal	0	2
16	sio	0	3

PM/0 State: Execute LIT

$sp \leftarrow sp + 1;$
 $stack[sp] \leftarrow M;$

Reg	Value	Stack	Value
IR	LIT 0 13	1	0
PC	4	2	0
BP	7	3	0
SP	13	4	0
		5	3
		6	0
		7	0
		8	1
		9	1
		10	16
		11	0
		12	0
		13	13
		14	0
		15	0

Line	OP	L	M
0	jmp	0	10
1	jmp	0	2
2	inc	0	6
3	lit	0	13
4	sto	0	4
5	lit	0	1
6	sto	1	4
7	lit	0	7
8	sto	0	5
9	opr	0	0
10	inc	0	6
11	lit	0	3
12	sto	0	4
13	lit	0	0
14	sto	0	5
15	cal	0	2
16	sio	0	3

PM/0 State: Execute STO

stack[base(L)+M] \leftarrow stack[sp];
sp \leftarrow sp - 1;

Reg	Value	Stack	Value
IR	STO 0 4	1	0
PC	5	2	0
BP	7	3	0
SP	12	4	0
		5	3
		6	0
		7	0
		8	1
		9	1
		10	16
		11	13
		12	0
		13	0
		14	0
		15	0

Line	OP	L	M
0	jmp	0	10
1	jmp	0	2
2	inc	0	6
3	lit	0	13
4	sto	0	4
5	lit	0	1
6	sto	1	4
7	lit	0	7
8	sto	0	5
9	opr	0	0
10	inc	0	6
11	lit	0	3
12	sto	0	4
13	lit	0	0
14	sto	0	5
15	cal	0	2
16	sio	0	3

PM/0 State: Execute LIT

$sp \leftarrow sp + 1;$
 $stack[sp] \leftarrow M;$

Reg	Value	Stack	Value
IR	LIT 0 1	1	0
PC	6	2	0
BP	7	3	0
SP	13	4	0
		5	3
		6	0
		7	0
		8	1
		9	1
		10	16
		11	13
		12	0
		13	1
		14	0
		15	0

Line	OP	L	M
0	jmp	0	10
1	jmp	0	2
2	inc	0	6
3	lit	0	13
4	sto	0	4
5	lit	0	1
6	sto	1	4
7	lit	0	7
8	sto	0	5
9	opr	0	0
10	inc	0	6
11	lit	0	3
12	sto	0	4
13	lit	0	0
14	sto	0	5
15	cal	0	2
16	sio	0	3

PM/0 State: Execute STO

$\text{stack}[\text{base}(\text{L})+\text{M}] \leftarrow \text{stack}[\text{sp}];$
 $\text{sp} \leftarrow \text{sp} - 1;$

Reg	Value	Stack	Value
IR	STO 1 4	1	0
PC	7	2	0
BP	7	3	0
SP	12	4	0
		5	1
		6	0
		7	0
		8	1
		9	1
		10	16
		11	13
		12	0
		13	0
		14	0
		15	0

Line	OP	L	M
0	jmp	0	10
1	jmp	0	2
2	inc	0	6
3	lit	0	13
4	sto	0	4
5	lit	0	1
6	sto	1	4
7	lit	0	7
8	sto	0	5
9	opr	0	0
10	inc	0	6
11	lit	0	3
12	sto	0	4
13	lit	0	0
14	sto	0	5
15	cal	0	2
16	sio	0	3

PM/0 State: Execute LIT

$sp \leftarrow sp + 1;$
 $stack[sp] \leftarrow M;$

Reg	Value	Stack	Value
IR	LIT 0 7	1	0
PC	8	2	0
BP	7	3	0
SP	13	4	0
		5	1
		6	0
		7	0
		8	1
		9	1
		10	16
		11	13
		12	0
		13	7
		14	0
		15	0

Line	OP	L	M
0	jmp	0	10
1	jmp	0	2
2	inc	0	6
3	lit	0	13
4	sto	0	4
5	lit	0	1
6	sto	1	4
7	lit	0	7
8	sto	0	5
9	opr	0	0
10	inc	0	6
11	lit	0	3
12	sto	0	4
13	lit	0	0
14	sto	0	5
15	cal	0	2
16	sio	0	3

PM/0 State: Execute STO

$\text{stack}[\text{base}(\text{L})+\text{M}] \leftarrow \text{stack}[\text{sp}];$
 $\text{sp} \leftarrow \text{sp} - 1;$

Reg	Value	Stack	Value
IR	STO 0 5	1	0
PC	9	2	0
BP	7	3	0
SP	12	4	0
		5	1
		6	0
		7	0
		8	1
		9	1
		10	16
		11	13
		12	7
		13	0
		14	0
		15	0

Line	OP	L	M
0	jmp	0	10
1	jmp	0	2
2	inc	0	6
3	lit	0	13
4	sto	0	4
5	lit	0	1
6	sto	1	4
7	lit	0	7
8	sto	0	5
9	opr	0	0
10	inc	0	6
11	lit	0	3
12	sto	0	4
13	lit	0	0
14	sto	0	5
15	cal	0	2
16	sio	0	3

PM/0 State: Fetch

Reg	Value	Stack	Value
IR	OPR 0 0	1	0
PC	10	2	0
BP	7	3	0
SP	12	4	0
		5	1
		6	0
		7	0
		8	1
		9	1
		10	16
		11	13
		12	7
		13	0
		14	0
		15	0

Line	OP	L	M
0	jmp	0	10
1	jmp	0	2
2	inc	0	6
3	lit	0	13
4	sto	0	4
5	lit	0	1
6	sto	1	4
7	lit	0	7
8	sto	0	5
9	opr	0	0
10	inc	0	6
11	lit	0	3
12	sto	0	4
13	lit	0	0
14	sto	0	5
15	cal	0	2
16	sio	0	3

PM/0 State: Execute OPR 0 (1/2: Pre-Stack Cleanup)

$sp \leftarrow bp - 1;$
 $pc \leftarrow stack[sp + 4];$
 $bp \leftarrow stack[sp + 3];$

Reg	Value	Stack	Value
IR	OPR 0 0	1	0
PC	16	2	0
BP	1	3	0
SP	6	4	0
		5	1
		6	0
		7	0
		8	1
		9	1
		10	16
		11	13
		12	7
		13	0
		14	0
		15	0

Line	OP	L	M
0	jmp	0	10
1	jmp	0	2
2	inc	0	6
3	lit	0	13
4	sto	0	4
5	lit	0	1
6	sto	1	4
7	lit	0	7
8	sto	0	5
9	opr	0	0
10	inc	0	6
11	lit	0	3
12	sto	0	4
13	lit	0	0
14	sto	0	5
15	cal	0	2
16	sio	0	3

PM/0 State: Execute OPR 0 (2/2: Post-Stack Cleanup)

$sp \leftarrow bp - 1;$
 $pc \leftarrow stack[sp + 4];$
 $bp \leftarrow stack[sp + 3];$

Reg	Value	Stack	Value
IR	OPR 0 0	1	0
PC	16	2	0
BP	1	3	0
SP	6	4	0
		5	1
		6	0
		7	0
		8	0
		9	0
		10	0
		11	0
		12	0
		13	0
		14	0
		15	0

Line	OP	L	M
0	jmp	0	10
1	jmp	0	2
2	inc	0	6
3	lit	0	13
4	sto	0	4
5	lit	0	1
6	sto	1	4
7	lit	0	7
8	sto	0	5
9	opr	0	0
10	inc	0	6
11	lit	0	3
12	sto	0	4
13	lit	0	0
14	sto	0	5
15	cal	0	2
16	sio	0	3

PM/0 State: Execute SIO 0,3

halt;

Reg	Value	Stack	Value
IR	SIO 0, 3	1	0
PC	0	2	0
BP	0	3	0
SP	0	4	0
		5	0
		6	0
		7	0
		8	0
		9	0
		10	0
		11	0
		12	0
		13	0
		14	0
		15	0

Line	OP	L	M
0	jmp	0	10
1	jmp	0	2
2	inc	0	6
3	lit	0	13
4	sto	0	4
5	lit	0	1
6	sto	1	4
7	lit	0	7
8	sto	0	5
9	opr	0	0
10	inc	0	6
11	lit	0	3
12	sto	0	4
13	lit	0	0
14	sto	0	5
15	cal	0	2
16	sio	0	3

Running a Program: Now With 80% More Making Sense

Initial values				pc	bp	sp	stack
				0	1	0	
0	jmp	0	10	10	1	0	
10	inc	0	6	11	1	6	0 0 0 0 0 0
11	lit	0	3	12	1	7	0 0 0 0 0 0 3
12	sto	0	4	13	1	6	0 0 0 0 3 0
13	lit	0	0	14	1	7	0 0 0 0 3 0 0
14	sto	0	5	15	1	6	0 0 0 0 3 0
15	cal	0	2	2	7	6	0 0 0 0 3 0
2	inc	0	6	3	7	12	0 0 0 0 3 0 0 1 1 16 0 0
3	lit	0	13	4	7	13	0 0 0 0 3 0 0 1 1 16 0 0 13
4	sto	0	4	5	7	12	0 0 0 0 3 0 0 1 1 16 13 0
5	lit	0	1	6	7	13	0 0 0 0 3 0 0 1 1 16 13 0 1
6	sto	1	4	7	7	12	0 0 0 0 1 0 0 1 1 16 13 0
7	lit	0	7	8	7	13	0 0 0 0 1 0 0 1 1 16 13 0 7
8	sto	0	5	9	7	12	0 0 0 0 1 0 0 1 1 16 13 7
9	opr	0	0	16	1	6	0 0 0 0 1 0
16	sio	0	3	0	0	0	