CS101 Computer Programming and Utilization

Milind Sohoni

May 4, 2006

In Summary

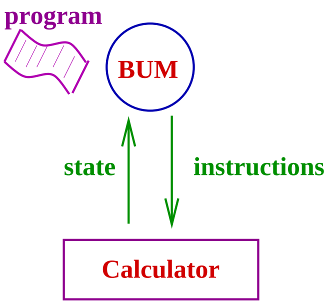
The Programmer

- Writes a CAL-program by assuming a typical input.
- Writes where typical inputs are to be replaced by user inputs.
- Stores/writes and transmits.

The Bum

- Receives the program.
- Substitutes his inputs.
- Runs the program on his calculator line-by-line in that order.

```
10   % substitute input here
*
8
DIV
5
+
32
=    % see output in display
```



Another Problem

Quadratic

Write a CAL-program to solve the quadratic $Ax^2 + Bx + C = 0$.

We solve a particular case and annotate the program in the right places. We consider $x^2 + 3x + 2$, thus $A = 1, B = 3, C = 2$.

- Compute $\Delta = \sqrt{B^2 - 4AC}$.
- Now calculate the two roots using the expressionss

$$roots = \frac{-B + \Delta}{2A} \quad and \quad \frac{B + \Delta}{2A}$$

```
3   % substitute B here
STO
RCL
*
RCL
-
4
*
1   % substitute A here
*
2   % substitute C here
=
SQRT
STO   % discriminant stored
```

We solve a particular case and annotate the program in the right places. We consider $x^2 + 3x + 2$, thus $A = 1, B = 3, C = 2$.

- Compute $\sqrt{B^2 - 4AC}$.
- Now calculate the two roots:

```
RCL
-
3    % substitute B here
=
DIV
2
DIV
1    % substitute A here
=    % read root 1 here
```

Clunky

- We see that there are repeated insertions of A, B, C. This is clumsy.
- We allow our calculator to have more than 1 memory register.
- `STO 5` will mean `store contents of visible register in memory location 5.`
- `RCL 5` will mean `move contents of memory location 5 to the visible register.`

Thus the set of instructions will now be:

- `+,-,=,*,DIV`
- `AC`
- `numbers`
- `STO 1, STO 2 ,...,STO 10`
- `RCL 1, RCL 2 ,...,RCL 10`

Programs in this language will be called MCAL-programs.

Final Code

```
1      % subs. A here
STO 1 % A is in M1
3      % subs B here
STO 2 % B in M2
2      % C
STO 3 % in M3
RCL 2
*
RCL 2
-
4
*
RCL 1
*
RCL 3
=
SQRT
STO 4 % disc in M4
```

```
RCL 4
-
RCL 2
=
DIV
2
DIV
RCL 1
=
STO 5  % first root
```

KEY

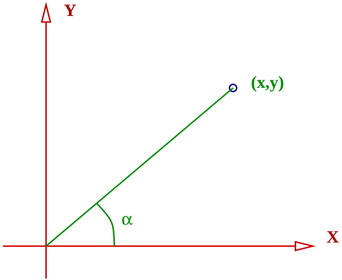
M1	A
M2	B
M3	C
M5	root 1
M6	root 2

Another example: Computing Slope

We are given a point (x, y). We wish to compute $\sin\alpha$ where α is the angle between the X-axis and the line joining the origin.

$$\sin\alpha = \frac{y}{\sqrt{x^2 + y^2}}$$

- first compute $\sqrt{x^2 + y^2}$.
- next compute $\frac{y}{\sqrt{x^2+y^2}}$.



We pick typical x, y say $(3, 4)$.

We pick typical x, y say $(3, 4)$.

```
3     % this is x
STO 1 %
4     % this is y
STO 2 %
*
RCL 2
+
RCL 1
*
RCL 1
=
SQRT
STO 4 % the denominator
RCL 2
DIV
RCL 4
=
STO 3 % answer stored in M3
```

Here:

M1	x	input
M2	y	input
M3	sin alpha	output

We pick typical x, y say $(3, 4)$.

```
3     % this is x
STO 1 %
4     % this is y
STO 2 %
*
RCL 2
+
RCL 1
*
RCL 1
=
SQRT
STO 4 % the denominator
RCL 2
DIV
RCL 4
=
STO 3 % answer stored in M3
```

Here:

M1	x	input
M2	y	input
M3	sin alpha	output

Summary

- Enhance calculator with more memory.
- Use STO 5, RCL 5 as additional instructions.
- Groups registers as Input, Output and temporary.

The Fibonacci numbers

Let $a_0 = 1$ and $a_1 = 1$, and a_n be given by the following recursive definition:

$$a_n = a_{n-1} + a_{n-2}$$

Write an MCAL-program to compute a_n.

Thus:

a_0	1
a_1	1
a_2	2
a_3	3
a_4	5
a_5	8
$\vdots$	$\vdots$

The Fibonacci numbers

Let $a_0 = 1$ and $a_1 = 1$, and a_n be given by the following recursive definition:

$$a_n = a_{n-1} + a_{n-2}$$

Write an MCAL-program to compute a_n.

```
4   %value of n
1
STO 1 % will store a(n-1)
STO 2 % will store a(n-2)

RCL 1 % beginning of loop
+
RCL 2
=
STO 3 % temporary
RCL 1
STO 2
RCL 3
STO 1 % end of loop
```

Our variable storage is as follows:

M1	a_{n-1}
M2	a_{n-2}

The Fibonacci numbers

Let $a_0 = 1$ and $a_1 = 1$, and a_n be given by the following recursive definition:

$$a_n = a_{n-1} + a_{n-2}$$

Write an MCAL-program to compute a_n.

```
4   %value of n
1
STO 1 % will store a(n-1)
STO 2 % will store a(n-2)

RCL 1 % beginning of loop
+
RCL 2
=
STO 3 % temporary
RCL 1
STO 2
RCL 3
STO 1 % end of loop
```

Our variable storage is as follows:

M1	a_{n-1}
M2	a_{n-2}

Next,

- There is an initialization part.
- and then a loop.

The Fibonacci numbers

```
4   %value of n
1
STO 1 % will store a(n-1)
STO 2 % will store a(n-2)

RCL 1 % beginning of loop
+
RCL 2
=
STO 3 % temporary
RCL 1
STO 2
RCL 3
STO 1 % end of loop
```

Let us analyse the loop. Assume that M1 has stored a_{n-1} and M2 has stored a_{n-2}. We plot M1,M2,M3,D, the display register.

The Fibonacci numbers

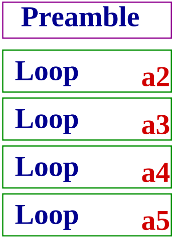
```
4   %value of n
1
STO 1 % will store a(n-1)
STO 2 % will store a(n-2)

RCL 1 % beginning of loop
+
RCL 2
=
STO 3 % temporary
RCL 1
STO 2
RCL 3
STO 1 % end of loop
```

Let us analyse the loop. Assume that M1 has stored a_{n-1} and M2 has stored a_{n-2}. We plot M1,M2,M3,D, the display register.

	M1	M2	M3	D
RCL 1	a_{n-1}	a_{n-2}	x	a_{n-1}
+	a_{n-1}	a_{n-2}	x	a_{n-1}
RCL 2	a_{n-1}	a_{n-2}	x	a_{n-2}
=	a_{n-1}	a_{n-2}	x	a_n
STO 3	a_{n-1}	a_{n-2}	a_n	a_n
RCL 1	a_{n-1}	a_{n-2}	a_n	a_{n-1}
STO 2	a_{n-1}	a_{n-1}	a_n	a_{n-1}
RCL 3	a_{n-1}	a_{n-1}	a_n	a_n
STO 1	a_n	a_{n-1}	a_n	a_n

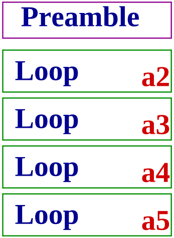
The Solution and ...

We see that the code consists of a preamble and a loop. The preamble sets up the recurrence relation and the loop executes it. The *loop* may be repeated as many times as required. Thus if $n = 5$, then a_0, a_1 are inputed raw, and the first loop calculates a_2. Thus the loop needs to be executed $n - 1$ times to compute a_n.



The Solution and ...

We see that the code consists of a preamble and a loop. The preamble sets up the recurrence relation and the loop executes it. The *loop* may be repeated as many times as required. Thus if $n = 5$, then a_0, a_1 are inputed raw, and the first loop calculates a_2. Thus the loop needs to be executed $n - 1$ times to compute a_n.



The Problem

But this NOT what we want.
The code changes with the input n, while we want it to be fixed!

What do we need?

Let us recall the BUM model of running programs. which goes through the following steps:

- We write a program and give it to BUM.
- BUM executes the first each instruction on the page sequentially.

Thus, in other words, the BUM may as well eat up each line since he will never use it again. From our earlier loop program, we see that there are some set of instructions which need to be executed again and again. This is the key observation.

What do we need?

Let us recall the BUM model of running programs. which goes through the following steps:

- We write a program and give it to BUM.
- BUM executes the first each instruction on the page sequentially.

Thus, in other words, the BUM may as well eat up each line since he will never use it again. From our earlier loop program, we see that there are some set of instructions which need to be executed again and again. This is the key observation.

- Let us devise a slightly cleverer DUMMY who retains the paper on which the program is written.

What do we need?

Let us recall the BUM model of running programs. which goes through the following steps:

- We write a program and give it to BUM.
- BUM executes the first each instruction on the page sequentially.

Thus, in other words, the BUM may as well eat up each line since he will never use it again. From our earlier loop program, we see that there are some set of instructions which need to be executed again and again. This is the key observation.

- Let us devise a slightly cleverer DUMMY who retains the paper on which the program is written.
- Let us add an instruction which alerts the DUMMY.

The new format

Our program will now have a line number. Thus each line has a number and a CALC-instruction. For example:

1	10	% substitute input here
2	*	
3	8	
4	DIV	
5	5	
6	+	
7	32	
8	=	% see output in display

The new format

Our program will now have a line number. Thus each line has a number and a CALC-instruction. For example:

1	10	% substitute input here
2	*	
3	8	
4	DIV	
5	5	
6	+	
7	32	
8	=	% see output in display

We add a new instruction:

TEST `nos`

This is executed by the DUMMY as follows.

- On encountering the TEST instruction, the DUMMY scans the DISPLAY.
- If the Display holds a negative or zero value, the DUMMY moves to the next instruction.
- If, however, the Display value is positive, the DUMMY moves to line number nos.

The Fibonacci Program

		initialization
1	5	this is n
	$STO5$	
	$-$	
	1	
	$=$	
	$STO4$	the counter
	1	a_{n-1} and a_{n-2}
	$STO1$	
9	$STO2$	

This completes the initialization.
At the end of this phase, we have:

M1	1
M2	1
M4	$4 = (n-1)$
M5	$5 = (n)$

The Fibonacci Program

	initialization
1	5 this is n
	$STO5$
	$-$
	1
	$=$
	$STO4$ the counter
	1 a_{n-1} and a_{n-2}
	$STO1$
9	$STO2$

This completes the initialization.
At the end of this phase, we have:

M1	1
M2	1
M4	$4 = (n-1)$
M5	$5 = (n)$

10	$RCL1$ the loop
	$+$
	$RCL2$
	$=$
	$STO3$ temporary
	$RCL1$
	$STO2$
	$RCL3$
18	$STO1$
19	$RCL4$ the termination
	$-$
	1
	$=$
	$STO4$ counter -1
24	$TEST$ 10
25	$STOP$

The Fibonacci Program

10	*RCL*1	the loop
	+	
	*RCL*2	
	=	
	*STO*3	temporary
	*RCL*1	
	*STO*2	
	*RCL*3	
18	*STO*1	
19	*RCL*4	the termination
	−	
	1	
	=	
	*STO*4	counter -1
24	*TEST* 10	
25	*STOP*	

We observe the values of each register at various times:

line	10	18	24
M1	1	2	2
M2	1	1	1
M4	4	4	3

The Fibonacci Program

10	*RCL*1	the loop
	+	
	*RCL*2	
	=	
	*STO*3	temporary
	*RCL*1	
	*STO*2	
	*RCL*3	
18	*STO*1	
19	*RCL*4	the termination
	−	
	1	
	=	
	*STO*4	counter -1
24	*TEST* 10	
25	*STOP*	

We observe the values of each register at various times:

line	10	18	24
M1	1	2	2
M2	1	1	1
M4	4	4	3

display is positive, so next instruction is 10.

The Fibonacci Program

10	*RCL*1	the loop
	+	
	*RCL*2	
	=	
	*STO*3	temporary
	*RCL*1	
	*STO*2	
	*RCL*3	
18	*STO*1	
19	*RCL*4	the termination
	−	
	1	
	=	
	*STO*4	counter -1
24	*TEST* 10	
25	*STOP*	

We observe the values of each register at various times:

line	10	18	24
M1	1	2	2
M2	1	1	1
M4	4	4	3

display is positive, so next instruction is 10.

line	10	18	24
M1	2	3	3
M2	1	2	2
M4	3	3	2

The Fibonacci Program

10	*RCL*1	the loop
	+	
	*RCL*2	
	=	
	*STO*3	temporary
	*RCL*1	
	*STO*2	
	*RCL*3	
18	*STO*1	
19	*RCL*4	the termination
	−	
	1	
	=	
	*STO*4	counter -1
24	*TEST* 10	
25	*STOP*	

We observe the values of each register at various times:

line	10	18	24
M1	1	2	2
M2	1	1	1
M4	4	4	3

display is positive, so next instruction is 10.

line	10	18	24
M1	2	3	3
M2	1	2	2
M4	3	3	2

display is positive, so next instruction is 10.

The Fibonacci Program

10	*RCL*1	the loop
	+	
	*RCL*2	
	=	
	*STO*3	temporary
	*RCL*1	
	*STO*2	
	*RCL*3	
18	*STO*1	
19	*RCL*4	the termination
	−	
	1	
	=	
	*STO*4	counter -1
24	*TEST* 10	
25	*STOP*	

line	10	18	24
M1	3	5	5
M2	2	3	3
M4	2	2	1

The Fibonacci Program

10	*RCL*1	the loop
	+	
	*RCL*2	
	=	
	*STO*3	temporary
	*RCL*1	
	*STO*2	
	*RCL*3	
18	*STO*1	
19	*RCL*4	the termination
	−	
	1	
	=	
	*STO*4	counter -1
24	*TEST* 10	
25	*STOP*	

line	10	18	24
M1	3	5	5
M2	2	3	3
M4	2	2	1

display is positive, so next instruction is 10.

The Fibonacci Program

10	$RCL1$	the loop
	$+$	
	$RCL2$	
	$=$	
	$STO3$	temporary
	$RCL1$	
	$STO2$	
	$RCL3$	
18	$STO1$	
19	$RCL4$	the termination
	$-$	
	1	
	$=$	
	$STO4$	counter -1
24	$TEST$ 10	
25	$STOP$	

line	10	18	24
M1	3	5	5
M2	2	3	3
M4	2	2	1

display is positive, so next instruction is 10.

line	10	18	24
M1	5	8	8
M2	3	5	5
M4	1	1	0

The Fibonacci Program

10	$RCL1$	the loop
	$+$	
	$RCL2$	
	$=$	
	$STO3$	temporary
	$RCL1$	
	$STO2$	
	$RCL3$	
18	$STO1$	
19	$RCL4$	the termination
	$-$	
	1	
	$=$	
	$STO4$	counter -1
24	$TEST$ 10	
25	$STOP$	

line	10	18	24
M1	3	5	5
M2	2	3	3
M4	2	2	1

display is positive, so next instruction is 10.

line	10	18	24
M1	5	8	8
M2	3	5	5
M4	1	1	0

display is zero, so next instruction is 26.

The Fibonacci Program

10	*RCL*1	the loop
	+	
	*RCL*2	
	=	
	*STO*3	temporary
	*RCL*1	
	*STO*2	
	*RCL*3	
18	*STO*1	
19	*RCL*4	the termination
	−	
	1	
	=	
	*STO*4	counter -1
24	*TEST* 10	
25	*STOP*	

line	10	18	24
M1	3	5	5
M2	2	3	3
M4	2	2	1

display is positive, so next instruction is 10.

line	10	18	24
M1	5	8	8
M2	3	5	5
M4	1	1	0

display is zero, so next instruction is 26.
computation stops.

Another Problem

The Log problem

Given a positive integer n, the largest k so that $10^k \leq n$.

What is the strategy?

- Maintain the input n in one register.
- Maintain k and 2^k in separate registers.
- Compute $n+1-2^k$. If positive, loop back.

M1	$n+1$
M2	k
M3	2^k

Another Problem

The Log problem

Given a positive integer n, the largest k so that $10^k \leq n$.

1	155	n here
	$+$	
	1	
	$=$	
	$STO1$	stored 156!
	0	
	$STO2$	stores k
	1	
	$STO3$	stores 2^k

nos	$RCL2$	
	$+$	
	1	
	$=$	
	$STO2$	incremented k
	$RCL3$	
	$*$	
	10	
	$=$	
	$STO3$	incremented 2^k
	$RCL1$	
	$-$	
	$RCL3$	
	$=$	This is $n+1-2^k$
	$TEST$ *nos*	
	$STOP$	