

STEPHEN F. AUSTIN STATE UNIVERSITY

CSC 214

LAB 5

SAMUEL JENTSCH

ACCOUNT NUMBER: CS214114

The purpose of this program is to take two numbers from the input register, cube each number, and find the difference between the two numbers. I decided to break this problem into separate pieces, and to begin with the subroutine for cubing the number.

Process for cubing a number:

Start out with number in AC.

Store the number (number to be cubed).

Cubing a number X can be achieved by adding X to itself X times to get X², then adding X² to itself X times, giving X³.

X = 5

5³ = 125 = X³

5+5+5+5+5= 25 = X²

25+25+25+25+25 = 125 = X³

Use a control variable beginning at 0. When the (control variable - X >= 0), the loop is finished. Repeat the above step with X² being added to itself 5 times.

```
LOAD X
LoopSquare, LOAD X2
ADD X
STORE X2
LOAD Ctr
ADD One
STORE Ctr
SUBT X
SKIPCOND 000 /<0
JUMP LoopCubeStart
JUMP LoopSquare
LoopCubeStart, Clear
STORE Ctr /Reset Ctr to 0
LoopCube, LOAD X3
ADD X2
STORE X3
LOAD Ctr
ADD One
STORE Ctr
SUBT X
SKIPCOND 000 /<0
JUMP EndCube
JUMP LoopCube
EndCube, Load X3
Output
HALT
```

```
X, DEC 5
X2, DEC 0
X3, DEC 0
Ctr, DEC 0
One, DEC 1
```

The above works to cube a single number. This can be used as the sub routine.

Next, write a program that takes in two numbers as input, and displays the difference between the two numbers. After this portion is complete, the subroutine for cubing the number should be able to be added.

The first number will be stored in Num1, the second in Num2. The difference will be stored in the AC using SUBT (LOAD Num1, SUBT Num2), and then output to the console using OUTPUT.

Input is gotten using INPUT, which loads the contents of the INPUT register into the AC. This input can be used to store a value in Num1, and then a value in Num2. The difference can then be found between the two numbers.

```
INPUT
STORE Num1
INPUT
STORE Num2
LOAD Num1
SUBT Num2
OUTPUT
HALT
Num1, DEC 0
Num2, DEC 0
```

The above works to subtract Num2 from Num1 where both Num1 and Num2 are input by the user.

Now these two portions need to be combined. The cube of Num1 will be found using the subroutine for cubing and stored in CubeNum1. The cube of Num2 will be found using the subroutine for cubing and stored in CubeNum2. CubeNum1 will then be loaded into the AC using LOAD and then CubeNum2 will be subtracted from the value in the AC (SUBT CubeNum2). This difference is then output using the OUTPUT command.

TRY 1:

```
INPUT
STORE Num1
JnS Subr
LOAD X3
STORE CubeNum1
INPUT
STORE Num2
```

```

JnS Subr
LOAD X3
STORE CubeNum2
LOAD CubeNum1
SUBT CubeNum2
OUTPUT
HALT
Num1, DEC 0
Num2, DEC 0
CubeNum1, DEC 0
CubeNum2, DEC 0
Subr, HEX 0 /Subroutine for cubing numbers
STORE X /Store value in X
LOAD Zero
STORE X2
STORE X3
LoopSquare, LOAD X2
ADD X
STORE X2
LOAD Ctr
ADD One
STORE Ctr
SUBT X
SKIPCOND 000 /<0
JUMP LoopCubeStart
JUMP LoopSquare
LoopCubeStart, Clear
STORE Ctr /Reset Ctr to 0
LoopCube, LOAD X3
ADD X2
STORE X3
LOAD Ctr
ADD One
STORE Ctr
SUBT X
SKIPCOND 000 /<0
JUMP EndCube
JUMP LoopCube
EndCube, Load X3
JUMP Subr
X, DEC 5
X2, DEC 0
X3, DEC 0
Ctr, DEC 0
One, DEC 1
Zero, DEC 0

```

The above seems to hard to tackle. So I decided to try getting a single number from input and cubing it using the cube routine already constructed:

```

INPUT
STORE Num1
STORE X
JnS SubrCube
LOAD X3
OUTPUT
HALT
Num1, DEC 0
SubrCube, DEC 0
LOAD X
LoopSquare, LOAD X2
ADD X
STORE X2
LOAD Ctr
ADD One
STORE Ctr
SUBT X
SKIPCOND 000 /<0
JUMP LoopCubeStart
JUMP LoopSquare
LoopCubeStart, Clear
STORE Ctr /Reset Ctr to 0
LoopCube, LOAD X3
ADD X2
STORE X3
LOAD Ctr
ADD One
STORE Ctr
SUBT X
SKIPCOND 000 /<0
JUMP EndCube
JUMP LoopCube
EndCube, Load X3
JUMPI SubrCube
X, DEC 0
X2, DEC 0
X3, DEC 0
Ctr, DEC 0
One, DEC 1

```

This works for the behavior desired. My error was using JUMP instead of JUMPI to return to the calling code from the subroutine.

The following code works to load both numbers and call the cube subroutine with each number. There is currently an error in that the first number is cubed, but the second number is only squared by the subroutine:

```

INPUT

```

STORE Num1
STORE X
JnS SubrCube
LOAD X3
STORE CubeNum1
INPUT
STORE Num2
STORE X
JnS SubrCube
LOAD X3
STORE CubeNum2
LOAD CubeNum1
SUBT CubeNum2
OUTPUT
HALT
Num1, DEC 0
Num2, DEC 0
CubeNum1, DEC 0
CubeNum2, DEC 0
SubrCube, DEC 0
LOAD Zero
STORE X2
STORE X3
LOAD X
LoopSquare, LOAD X2
ADD X
STORE X2
LOAD Ctr
ADD One
STORE Ctr
SUBT X
SKIPCOND 000 /<0
JUMP LoopCubeStart
JUMP LoopSquare
LoopCubeStart, Clear
STORE Ctr /Reset Ctr to 0
LoopCube, LOAD X3
ADD X2
STORE X3
LOAD Ctr
ADD One
STORE Ctr
SUBT X
SKIPCOND 000 /<0
JUMP EndCube
JUMP LoopCube
EndCube, Load X3
OUTPUT
JUMPI SubrCube

X, DEC 0
X2, DEC 0
X3, DEC 0
Ctr, DEC 0
One, DEC 1
Zero, DEC 0

The following code takes a number as input, then cubes and stores the number. It then takes a second number as input, and then cubes and stores the number. It then loads the first number (cubed) and subtracts the second cubed number from it. The difference is then displayed to the console.

```
INPUT
STORE Num1
STORE X
JnS SubrCube
LOAD X3
STORE CubeNum1
INPUT
STORE Num2
STORE X
JnS SubrCube
LOAD X3
STORE CubeNum2
LOAD CubeNum1
SUBT CubeNum2
OUTPUT
HALT
Num1, DEC 0
Num2, DEC 0
CubeNum1, DEC 0
CubeNum2, DEC 0
SubrCube, DEC 0
LOAD Zero
STORE X2
STORE X3
STORE Ctr
LOAD X
LoopSquare, LOAD X2
ADD X
STORE X2
LOAD Ctr
ADD One
STORE Ctr
SUBT X
SKIPCOND 000 /<0
JUMP LoopCubeStart
JUMP LoopSquare
LoopCubeStart, Clear
```

```

STORE Ctr /Reset Ctr to 0
LoopCube, LOAD X3
ADD X2
STORE X3
LOAD Ctr
ADD One
STORE Ctr
SUBT X
SKIPCOND 000 /<0
JUMP EndCube
JUMP LoopCube
EndCube, Load X3
OUTPUT
JUMPI SubrCube
X, DEC 0
X2, DEC 0
X3, DEC 0
Ctr, DEC 0
One, DEC 1
Zero, DEC 0

```

Now, optimizing the code to remove unnecessary labels and adding comments:

```

INPUT /Get first number
STORE X /Store in X to be used by the cubing subroutine
JnS SubrCube /Call the subroutine to cube the value in X
LOAD X3 /Load the value of X cubed by the subroutine
STORE CubeNum1 /Store this as the cube of the first number input
INPUT /Get the second number
STORE X /Store this number to be used by the cubing subroutine
JnS SubrCube /Call the subroutine to cube the number
LOAD X3 /Load the value of X cubed by the subroutine
STORE CubeNum2 /Store this value as the cube of the second number
LOAD CubeNum1 /Load the first number cubed
SUBT CubeNum2 /Subtract the second cubed number from it
OUTPUT /Display the difference
HALT /Stop the program
CubeNum1, DEC 0 /Used to hold the first number cubed
CubeNum2, DEC 0 /Used to hold the second number cubed
SubrCube, DEC 0 /The subroutine for cubing the number stored in X
LOAD Zero /Used to refresh values
STORE X2 /Refresh X2 (Number squared)
STORE X3 /Refresh X3 (Number cubed)
STORE Ctr /Refresh the value used to control the loop)
LOAD X /Load the number to be squared
LoopSquare, LOAD X2 /To square the number, add X to itself X times.
ADD X
STORE X2
LOAD Ctr /Increment the loop control variable

```



```

ADD One
STORE Ctr
SUBT X /Subtract X from the control variable. If it is not negative, stop the loop (loop X times)
SKIPCOND 000 /<0
JUMP LoopCubeStart /If the loop has executed X times, jump to the cubing loop
JUMP LoopSquare /Otherwise, loop again
LoopCubeStart, Clear /Clear the AC for the cube loop.
STORE Ctr /Reset Ctr to 0
LoopCube, LOAD X3 /Add X2 (X squared) to itself X times to get X cubed.
ADD X2
STORE X3
LOAD Ctr
ADD One
STORE Ctr
SUBT X /Subtract X from the control variable. If it is not negative, stop the loop (loop X times)
SKIPCOND 000 /<0
JUMP EndCube /If the loop has executed X times, stop the loop
JUMP LoopCube /Otherwise loop again.
EndCube, Load X3 /Loop has ended, show the cubed value
OUTPUT
JUMPI SubrCube /Jump to the calling code
X, DEC 0 /Number to be cubed
X2, DEC 0 /Number squared
X3, DEC 0 /Number cubed
Ctr, DEC 0 /Loop control variable
One, DEC 1 /Used to increment by 1
Zero, DEC 0 /Used to reset values

```

The above is the working program.

For a test run (using decimal numbers), the first number input is 5 and the second number input is 4. 5 cubed is 125 and 4 cubed is 64. The difference between the two numbers is 61:

Entering the first number, the correct cubed value is found:

	label	opcode	operand	hex
004		LOAD	X3	1024
005		OUTPUT		6000
006		HALT		7000
007	Num1	DEC	0	0000
008	SubrCube	DEC	0	0000
009		LOAD	X	1022
00A	LoopSqu...	LOAD	X2	1023
00B		ADD	X	3022
00C		STORE	X2	2023
00D		LOAD	Ctr	1025
00E		ADD	One	3026
00F		STORE	Ctr	2025

AC	007D	(Hex)
IR	7000	(Hex)
MAR	006	(Hex)
MBR	007D	(Hex)
PC	007	(Hex)
INP...	5	

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
000	5000	2007	2022	0008	1024	6000	7000	0005	0004	1022	1023	3022	2023	1025	3026	2025
010	4022	8000	9014	900A	A000	2025	1024	3023	2024	1025	3026	2025	4022	8000	9020	9016
020	1024	C008	0005	0019	007D	0005	0001	0000	0000	0000	0000	0000	0000	0000	0000	0000
030	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
070	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
080	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

Machine halted normally.

And entering the second number, the correct cube of 4 is found and the difference between 125 and 64 is displayed:

	label	opcode	operand	hex
008		LOAD	X3	1031
009		STORE	CubeNum2	200F
00A		LOAD	CubeNum1	100E
00B		SUBT	CubeNum2	400F
00C		OUTPUT		6000
00D		HALT		7000
00E	CubeNum1	DEC	0	0000
00F	CubeNum2	DEC	0	0000
010	SubrCube	DEC	0	0000
011		LOAD	Zero	1034
012		STORE	X2	2030
013		STORE	X3	2031

AC	003D	(Hex)
IR	7000	(Hex)
MAR	00D	(Hex)
MBR	0040	(Hex)
PC	00E	(Hex)
INP...	4	

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
000	5000	202F	0010	1031	200E	5000	202F	0010	1031	200F	100E	400F	6000	7000	007D	0040
010	0008	1034	2030	2031	2032	1030	302F	2030	1032	3033	2032	402F	8000	9020	C010	9016
020	A000	2032	1031	3030	2031	1032	3033	2032	402F	8000	902C	9022	1031	6000	C010	0004
030	0010	0040	0004	0001	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
040	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
050	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
060	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
070	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
080	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

Machine halted normally.

Here is the Symbol Table for the final program:

Symbol	Location
Ctr	032
CubeNum1	00E
CubeNum2	00F
EndCube	02C
LoopCube	022
LoopCubeStart	020
LoopSquare	016
One	033
SubrCube	010
X	02F
X2	030
X3	031
Zero	034

And the listing for the final program:

```

INPUT /Get first number
STORE X /Store in X to be used by the cubing subroutine
JnS SubrCube /Call the subroutine to cube the value in X
LOAD X3 /Load the value of X cubed by the subroutine
STORE CubeNum1 /Store this as the cube of the first number input
INPUT /Get the second number
STORE X /Store this number to be used by the cubing subroutine
JnS SubrCube /Call the subroutine to cube the number
LOAD X3 /Load the value of X cubed by the subroutine
STORE CubeNum2 /Store this value as the cube of the second number
LOAD CubeNum1 /Load the first number cubed
SUBT CubeNum2 /Subtract the second cubed number from it
OUTPUT /Display the difference
HALT /Stop the program
CubeNum1, DEC 0 /Used to hold the first number cubed
CubeNum2, DEC 0 /Used to hold the second number cubed
SubrCube, DEC 0 /The subroutine for cubing the number stored in X
LOAD Zero /Used to refresh values
STORE X2 /Refresh X2 (Number squared)

```

```

STORE X3 /Refresh X3 (Number cubed)
STORE Ctr /Refresh the value used to control the loop)
LOAD X /Load the number to be squared
LoopSquare, LOAD X2 /To square the number, add X to itself X times.
ADD X
STORE X2
LOAD Ctr /Increment the loop control variable
ADD One
STORE Ctr
SUBT X /Subtract X from the control variable. If it is not negative, stop the loop (loop X times)
SKIPCOND 000 /<0
JUMP LoopCubeStart /If the loop has executed X times, jump to the cubing loop
JUMP LoopSquare /Otherwise, loop again
LoopCubeStart, Clear /Clear the AC for the cube loop.
STORE Ctr /Reset Ctr to 0
LoopCube, LOAD X3 /Add X2 (X squared) to itself X times to get X cubed.
ADD X2
STORE X3
LOAD Ctr
ADD One
STORE Ctr
SUBT X /Subtract X from the control variable. If it is not negative, stop the loop (loop X times)
SKIPCOND 000 /<0
JUMP EndCube /If the loop has executed X times, stop the loop
JUMP LoopCube /Otherwise loop again.
EndCube, Load X3 /Loop has ended, show the cubed value
OUTPUT
JUMPI SubrCube /Jump to the calling code
X, DEC 0 /Number to be cubed
X2, DEC 0 /Number squared
X3, DEC 0 /Number cubed
Ctr, DEC 0 /Loop control variable
One, DEC 1 /Used to increment by 1
Zero, DEC 0 /Used to reset values

```