COSS Assignment 6

Batch: S1-18_DSEABZG516. Bangalore.

Group members: Pamidi Pradeep Kumar (2018AB04152), Bhagabat Prasad Bhuyan (2018AB04161)

Problem:

A data analyst Ms. Simran is working on information collected by a mobile service provider MARiO on data usage on monthly basis since 2016. The information collected is in the form of Matrix (Y axis - Total data downloaded in GB vs X axis - Month of the year). She is tasked to present a cumulative (successive additions) figure of data downloaded for each month in a year for the last two years i.e., 2017 and 2018. Implement the above using CPU OS simulator based assembly language program.

Source code for the program: https://github.com/pamidipradeep/CossAssignment6
Video of the running of the program: https://youtu.be/3oDOqxFIyyo A 2-minute video of the program running with 2 year data.
Please call Pradeep (+91-9886328027) if case you find any issues running the program

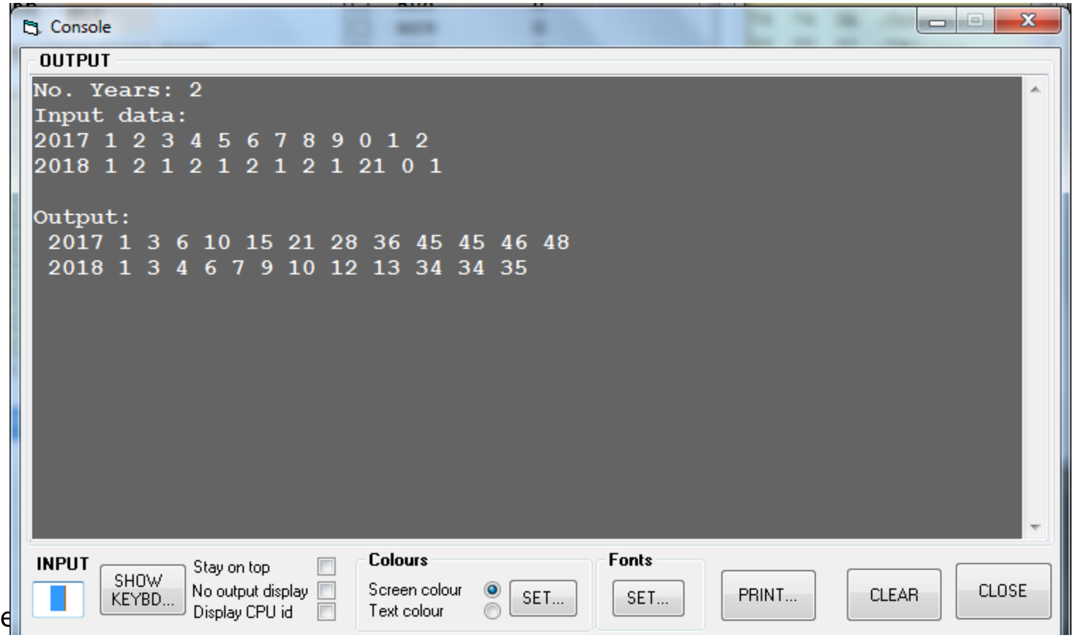
Key highlights of the program:

1. This program is designed to be flexible to work with any number of years of data.
2. This program accepts input from the keyboard in a user friendly manner.
3. This program outputs the data to the screen and saves the results to the memory too.
4. This program is **self-contained**. All the information, strings and I/O code are within the program itself and doesn't require any pre-memory data placement.

Note about input:

1. The values of the matrix of a row must be **space separated** (even the last element of the row will require a space after the value).
2. Enter key at the end of entering the row will take to the input of the next row.
3. First element of each row is the year for which the data is being entered.
4. To keep the focus on the core logic and to simplify the program, extensive validation like checking for special characters during input are not performed. So, please enter valid input and avoid backspace/alphabets/special-characters while entering the input.

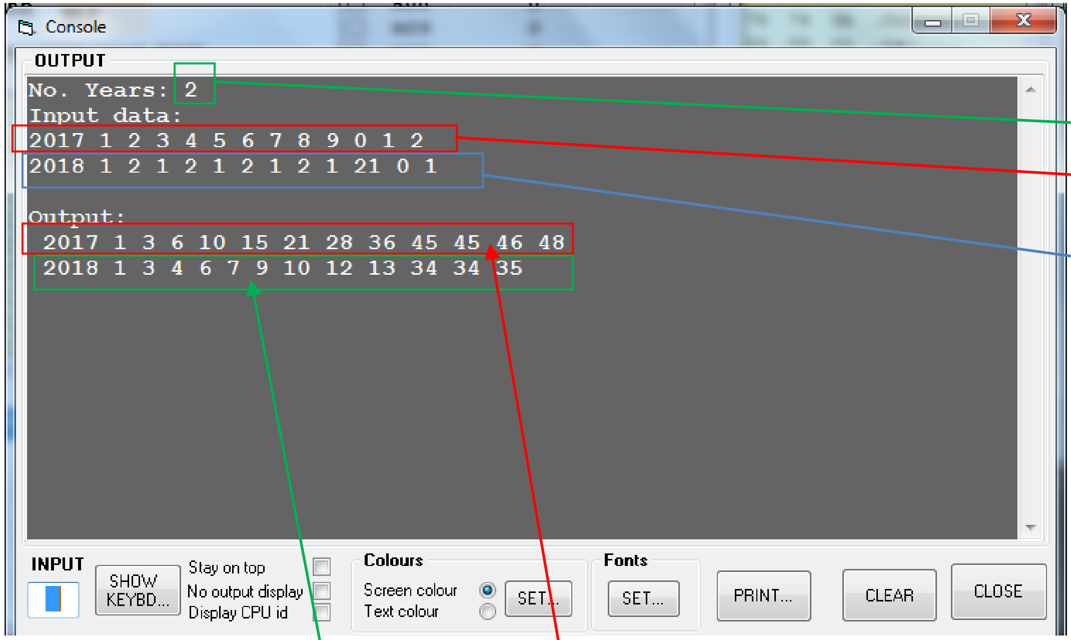
.



```
No. Years: 2
Input data:
2017 1 2 3 4 5 6 7 8 9 0 1 2
2018 1 2 1 2 1 2 1 2 1 21 0 1

Output:
 2017 1 3 6 10 15 21 28 36 45 45 46 48
 2018 1 3 4 6 7 9 10 12 13 34 34 35
```

Stay on top □
No output display □
Display CPU id □
Screen colour ●
Text colour ○

Input, Output and their mapping to the memory locations where the values are stored



Console

OUTPUT

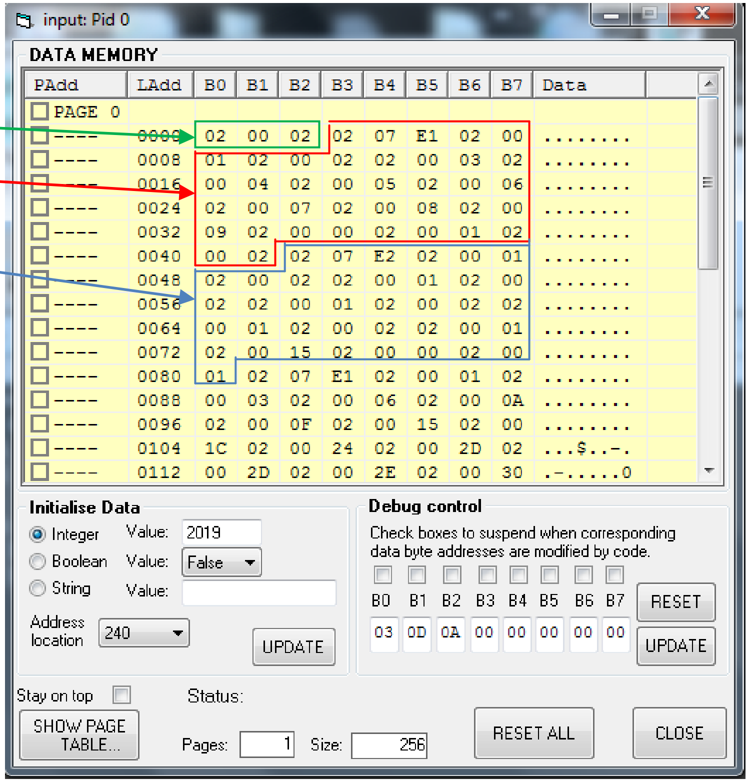
```
No. Years: 2
Input data:
2017 1 2 3 4 5 6 7 8 9 0 1 2
2018 1 2 1 2 1 2 1 2 1 21 0 1

Output:
 2017 1 3 6 10 15 21 28 36 45 45 46 48
 2018 1 3 4 6 7 9 10 12 13 34 34 35
```

INPUT

Stay on top □
No output display □
Display CPU id □

Colours
Screen colour ●
Text colour ○

Fonts



input: Pid 0

DATA MEMORY

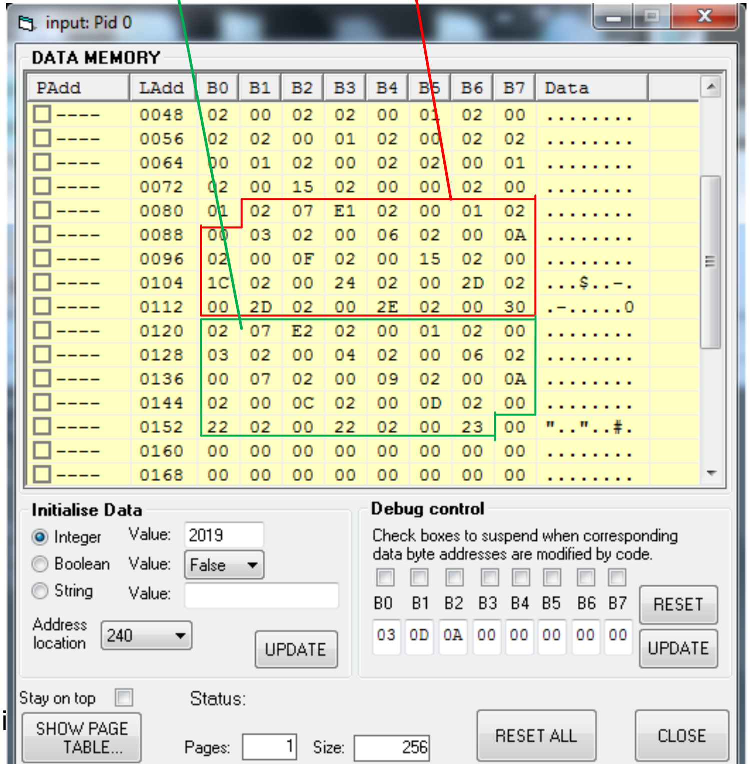
PAdd	LAdd	B0	B1	B2	B3	B4	B5	B6	B7	Data
□ PAGE 0										
□ ----	0000	02	00	02	02	07	E1	02	00	....
□ ----	0008	01	02	00	02	02	00	03	02	....
□ ----	0016	00	04	02	00	05	02	00	06	....
□ ----	0024	02	00	07	02	00	08	02	00	....
□ ----	0032	09	02	00	00	02	00	01	02	....
□ ----	0040	00	02	02	07	E2	02	00	01	....
□ ----	0048	02	00	02	02	00	01	02	00	....
□ ----	0056	02	02	00	01	02	00	02	02	....
□ ----	0064	00	01	02	00	02	02	00	01	....
□ ----	0072	02	00	15	02	00	00	02	00	....
□ ----	0080	01	02	07	E1	02	00	01	02	....
□ ----	0088	00	03	02	00	06	02	00	0A	....
□ ----	0096	02	00	0F	02	00	15	02	00	....
□ ----	0104	1C	02	00	24	02	00	2D	02	...$..-.
□ ----	0112	00	2D	02	00	2E	02	00	30	.-.....0

Initialise Data
● Integer Value: 2019
○ Boolean Value: False
○ String Value:
Address location 240

Debug control
Check boxes to suspend when corresponding data byte addresses are modified by code.

B0	B1	B2	B3	B4	B5	B6	B7
□	□	□	□	□	□	□	□
03	0D	0A	00	00	00	00	00

Stay on top □
Status:
Pages: 1 Size: 256



input: Pid 0

DATA MEMORY

PAdd	LAdd	B0	B1	B2	B3	B4	B5	B6	B7	Data
□ ----	0048	02	00	02	02	00	01	02	00	....
□ ----	0056	02	02	00	01	02	00	02	02	....
□ ----	0064	00	01	02	00	02	02	00	01	....
□ ----	0072	02	00	15	02	00	00	02	00	....
□ ----	0080	01	02	07	E1	02	00	01	02	....
□ ----	0088	00	03	02	00	06	02	00	0A	....
□ ----	0096	02	00	0F	02	00	15	02	00	....
□ ----	0104	1C	02	00	24	02	00	2D	02	...$..-.
□ ----	0112	00	2D	02	00	2E	02	00	30	.-.....0
□ ----	0120	02	07	E2	02	00	01	02	00	....
□ ----	0128	03	02	00	04	02	00	06	02	....
□ ----	0136	00	07	02	00	09	02	00	0A	....
□ ----	0144	02	00	0C	02	00	0D	02	00	....
□ ----	0152	22	02	00	22	02	00	23	00	"..."..#.
□ ----	0160	00	00	00	00	00	00	00	00	....
□ ----	0168	00	00	00	00	00	00	00	00	....

Initialise Data
● Integer Value: 2019
○ Boolean Value: False
○ String Value:
Address location 240

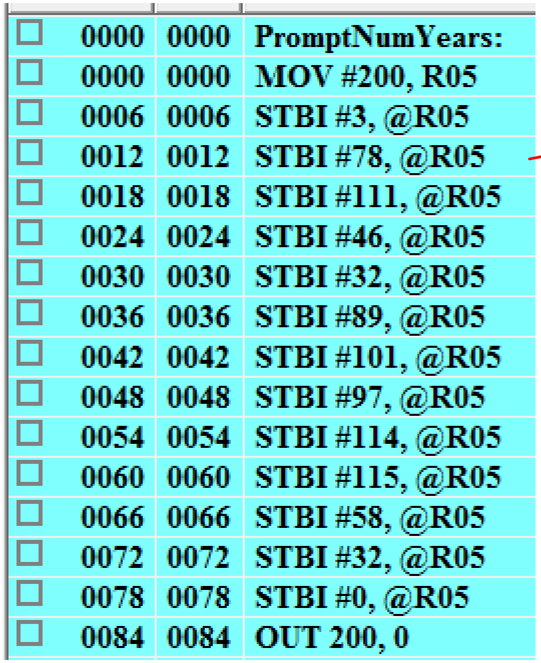
Debug control
Check boxes to suspend when corresponding data byte addresses are modified by code.

B0	B1	B2	B3	B4	B5	B6	B7
□	□	□	□	□	□	□	□
03	0D	0A	00	00	00	00	00

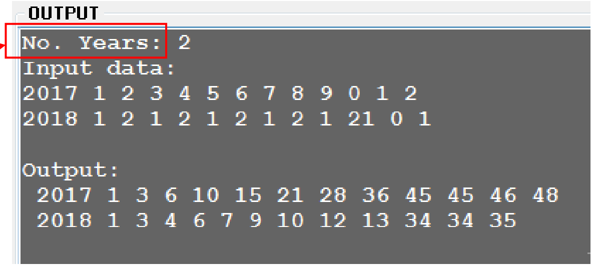
Stay on top □
Status:
Pages: 1 Size: 256

Program Segments and their brief description

1. **Display the message** and prompt the user to enter number of years: ASCII values of the message are moved to a memory location and then sent out to the output interface.

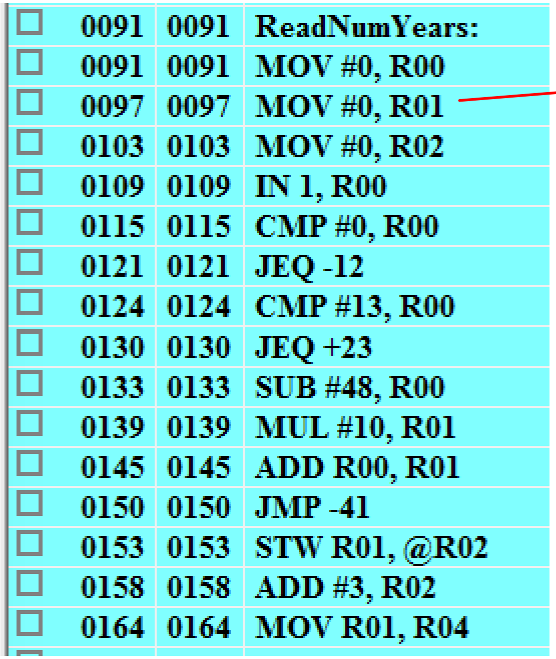


□	0000	0000	PromptNumYears:
□	0000	0000	MOV #200, R05
□	0006	0006	STBI #3, @R05
□	0012	0012	STBI #78, @R05
□	0018	0018	STBI #111, @R05
□	0024	0024	STBI #46, @R05
□	0030	0030	STBI #32, @R05
□	0036	0036	STBI #89, @R05
□	0042	0042	STBI #101, @R05
□	0048	0048	STBI #97, @R05
□	0054	0054	STBI #114, @R05
□	0060	0060	STBI #115, @R05
□	0066	0066	STBI #58, @R05
□	0072	0072	STBI #32, @R05
□	0078	0078	STBI #0, @R05
□	0084	0084	OUT 200, 0

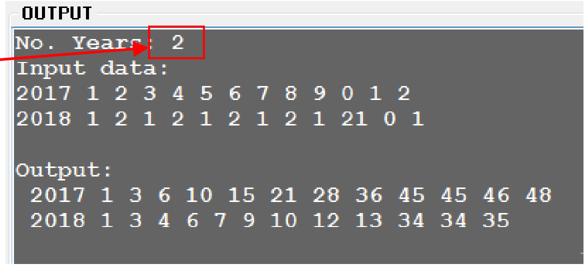


```
No. Years: 2
Input data:
2017 1 2 3 4 5 6 7 8 9 0 1 2
2018 1 2 1 2 1 2 1 2 1 21 0 1

Output:
 2017 1 3 6 10 15 21 28 36 45 45 46 48
 2018 1 3 4 6 7 9 10 12 13 34 34 35
```

2. **Accept the user input for the number of year** for which the data is being collected. The program is flexible to handle multiple year's data. Keep listening to the keyboard interface until a key is pressed. Once pressed, get the numeric value from the ASCII value received. This program can accept large numbers too.

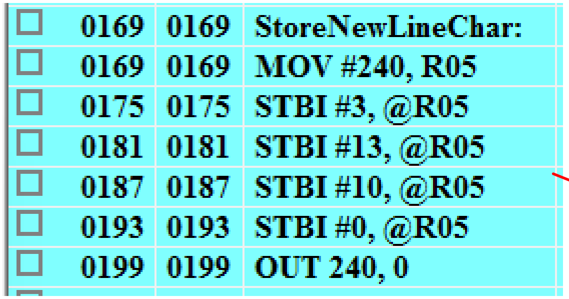


```
□ 0091 0091 ReadNumYears:
□ 0091 0091 MOV #0, R00
□ 0097 0097 MOV #0, R01
□ 0103 0103 MOV #0, R02
□ 0109 0109 IN 1, R00
□ 0115 0115 CMP #0, R00
□ 0121 0121 JEQ -12
□ 0124 0124 CMP #13, R00
□ 0130 0130 JEQ +23
□ 0133 0133 SUB #48, R00
□ 0139 0139 MUL #10, R01
□ 0145 0145 ADD R00, R01
□ 0150 0150 JMP -41
□ 0153 0153 STW R01, @R02
□ 0158 0158 ADD #3, R02
□ 0164 0164 MOV R01, R04
```



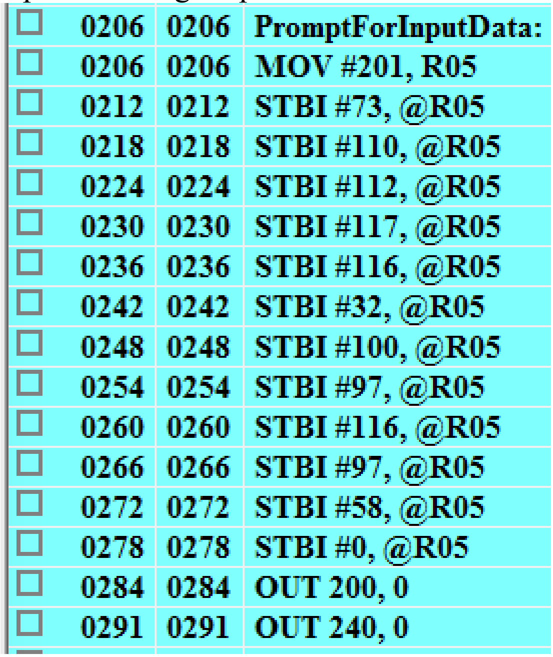
```
No. Years: 2
Input data:
2017 1 2 3 4 5 6 7 8 9 0 1 2
2018 1 2 1 2 1 2 1 2 1 21 0 1

Output:
 2017 1 3 6 10 15 21 28 36 45 45 46 48
 2018 1 3 4 6 7 9 10 12 13 34 34 35
```

3. Output a new line character to display the output in a different line. Store the new line character in a memory location to reuse it later.

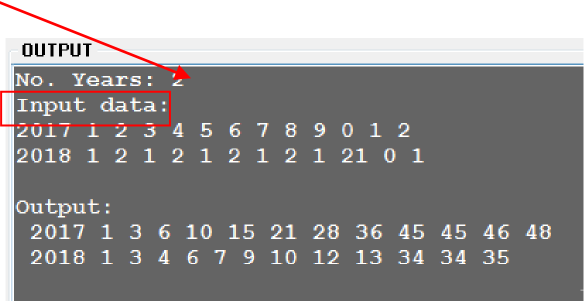


```
□ 0169 0169 StoreNewLineChar:
□ 0169 0169 MOV #240, R05
□ 0175 0175 STBI #3, @R05
□ 0181 0181 STBI #13, @R05
□ 0187 0187 STBI #10, @R05
□ 0193 0193 STBI #0, @R05
□ 0199 0199 OUT 240, 0
```

4. Output the string "Input data:" onto the screen:



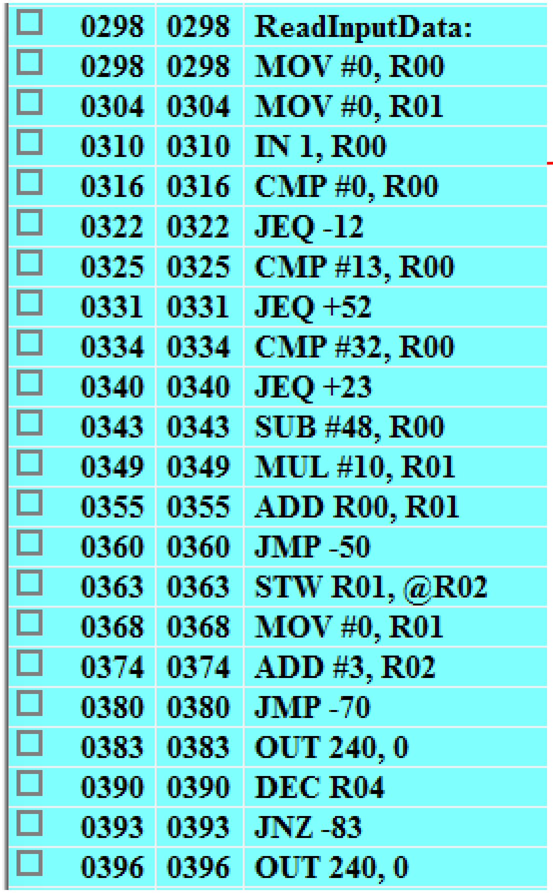
```
□ 0206 0206 PromptForInputData:
□ 0206 0206 MOV #201, R05
□ 0212 0212 STBI #73, @R05
□ 0218 0218 STBI #110, @R05
□ 0224 0224 STBI #112, @R05
□ 0230 0230 STBI #117, @R05
□ 0236 0236 STBI #116, @R05
□ 0242 0242 STBI #32, @R05
□ 0248 0248 STBI #100, @R05
□ 0254 0254 STBI #97, @R05
□ 0260 0260 STBI #116, @R05
□ 0266 0266 STBI #97, @R05
□ 0272 0272 STBI #58, @R05
□ 0278 0278 STBI #0, @R05
□ 0284 0284 OUT 200, 0
□ 0291 0291 OUT 240, 0
```



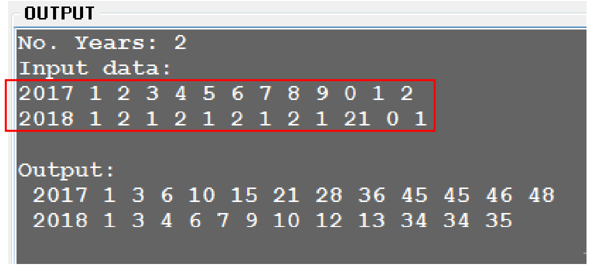
OUTPUT

```
No. Years: 2
Input data:
2017 1 2 3 4 5 6 7 8 9 0 1 2
2018 1 2 1 2 1 2 1 2 1 21 0 1

Output:
 2017 1 3 6 10 15 21 28 36 45 45 46 48
 2018 1 3 4 6 7 9 10 12 13 34 34 35
```

5. **Accept use input**. Read the keyboard and store the values entered in the memory. Each value is separated by a **space**. Enter key takes to the data of the next year.



```
□  0298  0298  ReadInputData:
□  0298  0298  MOV #0, R00
□  0304  0304  MOV #0, R01
□  0310  0310  IN 1, R00
□  0316  0316  CMP #0, R00
□  0322  0322  JEQ -12
□  0325  0325  CMP #13, R00
□  0331  0331  JEQ +52
□  0334  0334  CMP #32, R00
□  0340  0340  JEQ +23
□  0343  0343  SUB #48, R00
□  0349  0349  MUL #10, R01
□  0355  0355  ADD R00, R01
□  0360  0360  JMP -50
□  0363  0363  STW R01, @R02
□  0368  0368  MOV #0, R01
□  0374  0374  ADD #3, R02
□  0380  0380  JMP -70
□  0383  0383  OUT 240, 0
□  0390  0390  DEC R04
□  0393  0393  JNZ -83
□  0396  0396  OUT 240, 0
```

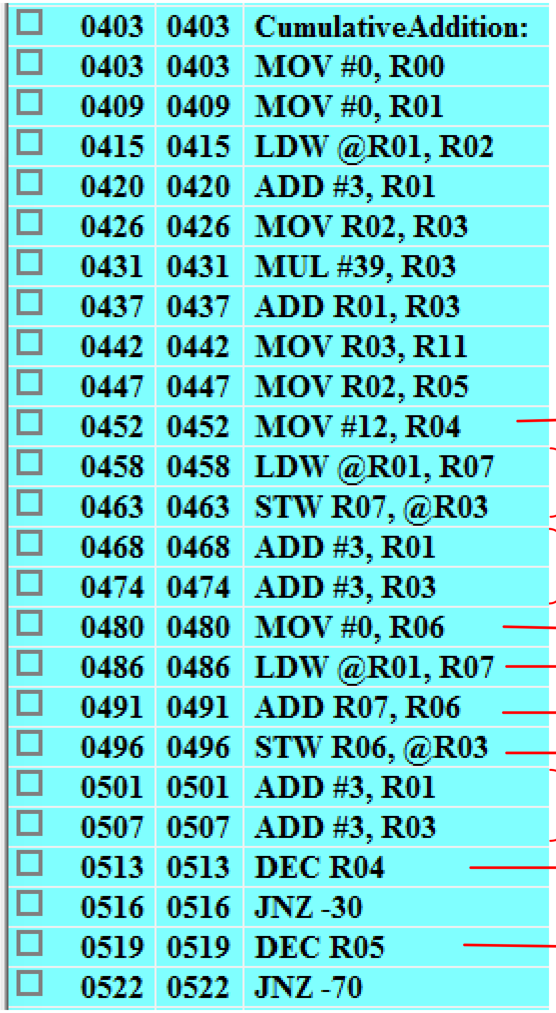


OUTPUT

```
No. Years: 2
Input data:
2017 1 2 3 4 5 6 7 8 9 0 1 2
2018 1 2 1 2 1 2 1 2 1 21 0 1

Output:
 2017 1 3 6 10 15 21 28 36 45 45 46 48
 2018 1 3 4 6 7 9 10 12 13 34 34 35
```

6. The main Business Logic:

1. Address pointers are stored in registers R01 (input values memory location) and R03 (output address)
2. Data of each month is read into register R07
3. The value is added to the cumulative sum being maintained in R06.
4. The value in R6 is stored into output memory location (address maintained in R03).
5. Registers R03 and R01 and incremented to point to the next value.
6. The above 4 steps are repeated 12 times. This completes the cumulative calculation for one year. Months counter is tracked in R04.
7. The above 5 steps are related for the number of years entered by the user. Years counter is stored in register R5



```
□  0403  0403  CumulativeAddition:
□  0403  0403  MOV #0, R00
□  0409  0409  MOV #0, R01
□  0415  0415  LDW @R01, R02
□  0420  0420  ADD #3, R01
□  0426  0426  MOV R02, R03
□  0431  0431  MUL #39, R03
□  0437  0437  ADD R01, R03
□  0442  0442  MOV R03, R11
□  0447  0447  MOV R02, R05
□  0452  0452  MOV #12, R04
□  0458  0458  LDW @R01, R07
□  0463  0463  STW R07, @R03
□  0468  0468  ADD #3, R01
□  0474  0474  ADD #3, R03
□  0480  0480  MOV #0, R06
□  0486  0486  LDW @R01, R07
□  0491  0491  ADD R07, R06
□  0496  0496  STW R06, @R03
□  0501  0501  ADD #3, R01
□  0507  0507  ADD #3, R03
□  0513  0513  DEC R04
□  0516  0516  JNZ -30
□  0519  0519  DEC R05
□  0522  0522  JNZ -70
```

Calculate the output address. The output starts from an address that's just after the input data.

Months counter. Inner loop executes 12 times

Move the year (first column) in the matrix to the output

Increment the address pointers to point to the next value

Set the cumulative value for the year to zero.

Load the value of the month.

Calculate the cumulative result by adding to the previous

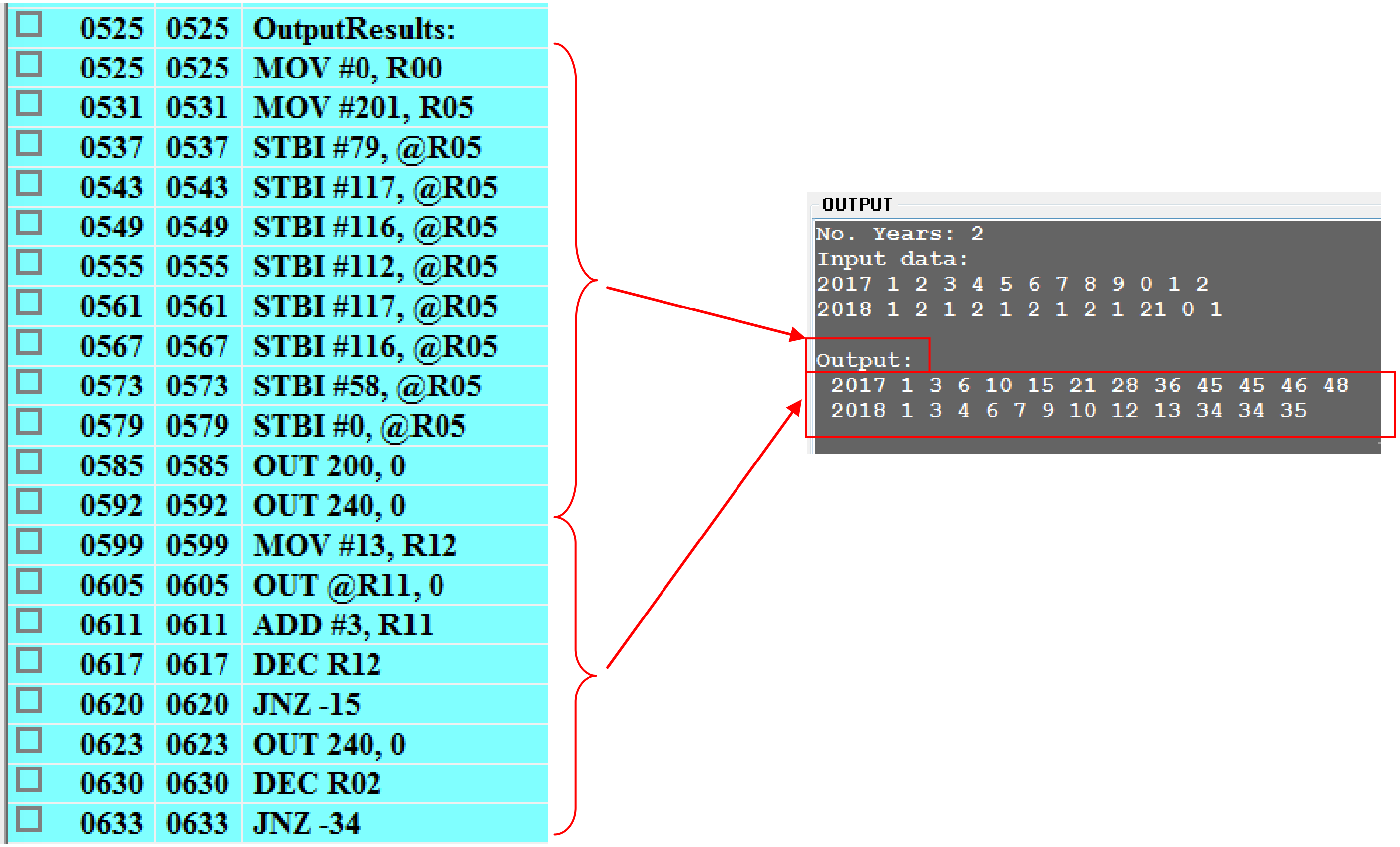
Store the result in the memory

Increment the address pointers (input and output) to point to the next value

Decrement the number of months counter.

Decrement the years counter. Outer loop.

7. Finally, **display the results** stored in the output memory location onto the screen. There are two loops: one for repeating the months (x-axis) and another for the number of years (y-axis).



```
□  0525  0525  OutputResults:
□  0525  0525  MOV #0, R00
□  0531  0531  MOV #201, R05
□  0537  0537  STBI #79, @R05
□  0543  0543  STBI #117, @R05
□  0549  0549  STBI #116, @R05
□  0555  0555  STBI #112, @R05
□  0561  0561  STBI #117, @R05
□  0567  0567  STBI #116, @R05
□  0573  0573  STBI #58, @R05
□  0579  0579  STBI #0, @R05
□  0585  0585  OUT 200, 0
□  0592  0592  OUT 240, 0
□  0599  0599  MOV #13, R12
□  0605  0605  OUT @R11, 0
□  0611  0611  ADD #3, R11
□  0617  0617  DEC R12
□  0620  0620  JNZ -15
□  0623  0623  OUT 240, 0
□  0630  0630  DEC R02
□  0633  0633  JNZ -34
```

```
No. Years: 2
Input data:
2017 1 2 3 4 5 6 7 8 9 0 1 2
2018 1 2 1 2 1 2 1 2 1 21 0 1

Output:
 2017 1 3 6 10 15 21 28 36 45 45 46 48
 2018 1 3 4 6 7 9 10 12 13 34 34 35
```