

### **Part I:**

Load ptr to first value in FAHTEMP

Load ptr to first value in FAHAVG

Load register with counter value

Repeat

    Load register with first FAHTEMP value, decrement by 1

    Add FAHTEMP value to holding register

    Decrement counter

Until counter = 0

Divide holding register by 16

Add carry if any to the holding register

Store the holder register

### **Part II - Attempt II:**

Pass in pointer to data and length using R0, R1

Save register to be used in program on the stack

Copy data ptr to secondary register (2)

Copy counter to secondary register (2)

Load first value from data ptr(1) into register, increment data pointer by 1

Subtract counter(1) by 1

Repeat

    Load next value from data ptr into second register, increment by 1

    Compare values

    Store the largest value back in first register

    Decrement counter 1

Until counter = 0

Load first value from data ptr(2) into register, increment data pointer by 1

Subtract counter(2) by 1

Repeat

    Load next value from data ptr into second register, increment by 1

Compare values  
Store the smallest value back in first register  
Decrement counter 1  
Until counter = 0  
Restore Stack  
Carry min max values out using R2 and R3

**Part II - Attempt III:**

Pass in pointer to data and length using R0, R1  
Save register to be used in program on the stack  
Load first value from data ptr into register, increment data pointer by 1  
Copy first value from data ptr into secondary register  
Subtract counter by 1  
Repeat  
    Load next value from data ptr into second register, increment by 1  
    Compare values  
    Store the largest value back in first register  
    Compare values  
    Store the smallest value back in the second register  
    Decrement counter 1  
Until counter = 0  
Restore Stack  
Carry min max values out using R2 and R3

## Breakdown of compare

R4 is always holding values from the data array, R2 holds the largest to be seen:

```
CMP R2, R4
```

```
MOVMI R2, R4
```

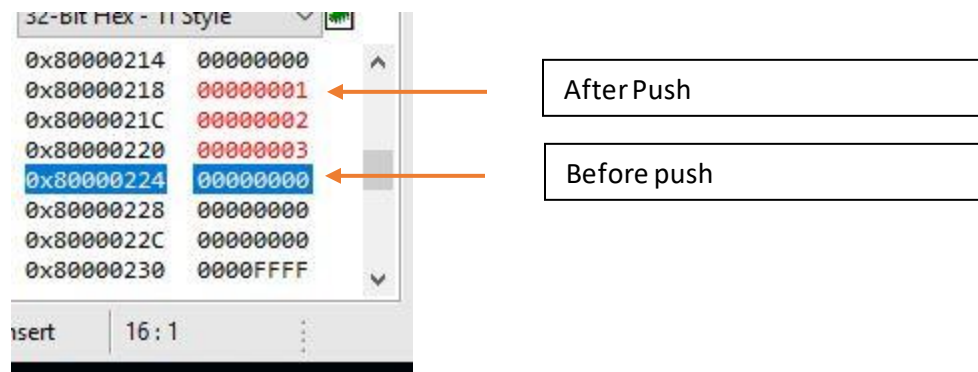
If  $R2 > R4$ , then  $Z = 0$ ,  $N = 0$ , MOVMI DOES NOT trigger, R2 is saved

If  $R2 < R4$ , then  $Z = 0$ ,  $N = 1$ , MOVMI triggers, R4 is saved over R2

If  $R2 = R4$ , then it doesn't matter if MOVMI occurs or not

This works the exact same way to save the smallest value, but PL is used instead which is the opposite of MI, so it saves all smaller values.

## Stack Breakdown



Here I tested the stack using the values 0x03, 0x02, 0x01.

```
R4 = 0x03
```

```
R1 = 0x02
```

```
R0 = 0x01
```

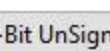
Before Push	0x224	0000000000	
	0x220	0x03	R4
	0x20c	0x02	R1
After Push	0x218	0x01	R2
	0x214	0000000000	

Stack started at 0x124, added 0x100, since I'm using FD, the first spot 0x224 is empty, and the stack ends up pointing to 0x218.

connected : Unknown)

0x8000010A	15
0x8000010B	16
0x8000010C	9
0x8000010D	16
0x8000010E	1
0x8000010F	0
0x80000110	0
0x80000111	0

Avg  
Max  
Min



0x80000104 - 0x80000108(-0x4...

8-Bit Unsigned Int

0x80000104 0

0x80000105 0

0x80000106 0

0x80000107 0

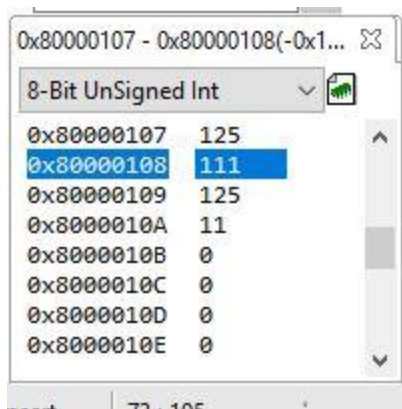
0x80000108 0

0x80000109 0

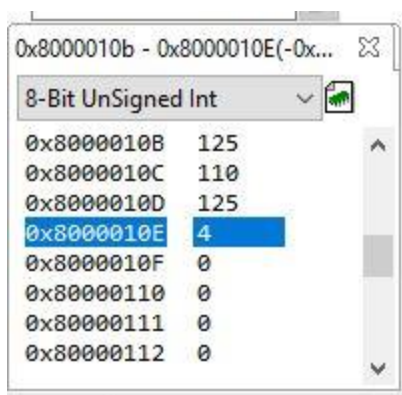
0x8000010A 0

0x8000010B 0

[illegible]



Test Values	125, 125, 125, 125, 125, 12, 125, 125, 125, 125, 125, 125, 11, 125, 125, 125	
	Expected Result	Program Output
Average	110.8	111
MAX	125	125
MIN	11	11



Test Values	125, 125, 125, 125, 125, 11, 125, 125, 125, 125, 125, 125, 4, 125, 125, 125	
	Expected Result	Program Output
Average	110.31	110
MAX	125	125
MIN	4	4