Jeffrey Razon

**CPE301 – SPRING 2018**

Since it has 16 bit to store the sum you need to get the 16 bit value. Design Assignment 01

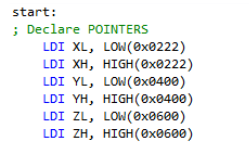
**DO NOT REMOVE THIS PAGE DURING SUBMISSION:**

The student understands that all required components should be submitted in complete for grading of this assignment.

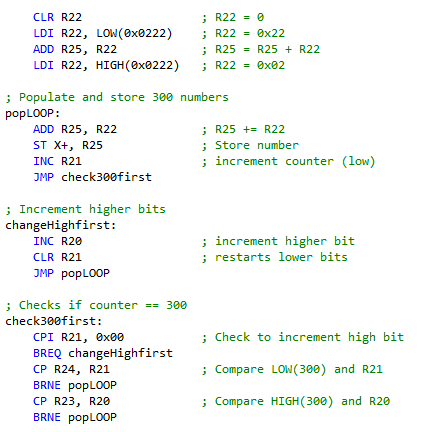
|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **SUBMISSION ITEM** | **COMPLETED (Y/N)** | **MARKS**  **(/MAX)** |
| 1 | COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS |  |  |
| 2. | INITIAL CODE OF TASK 1/A |  |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 2/B |  |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 3/C |  |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 4/D |  |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 5/E |  |  |
| 4. | SCHEMATICS |  |  |
| 5. | SCREENSHOTS OF EACH TASK OUTPUT |  |  |
| 5. | SCREENSHOT OF EACH DEMO |  |  |
| 6. | VIDEO LINKS OF EACH DEMO |  |  |
| 7. | GOOGLECODE LINK OF THE DA |  |  |
|  |  |  |  |
|  |  |  |  |

**Task 1/A**: Store 300 numbers starting from the STARTADDS=0x0222 location. Populate the value of the memory location by adding high(STARTADDS) and low(STARTADDS) . Use the X/Y/Z registers as pointers to fill up 300 numbers.

a) Declaring X/Y/Z registers as pointers

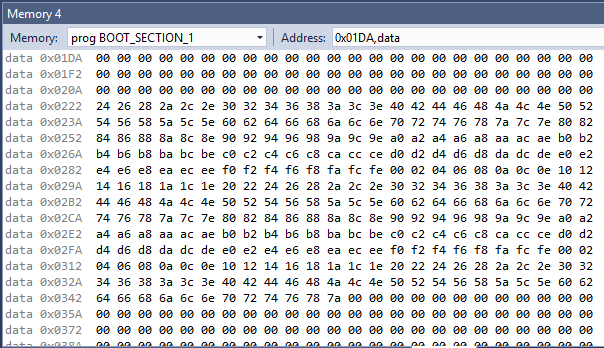


b) Populating 300 numbers and storing them (counter checks for exactly 300 numbers)

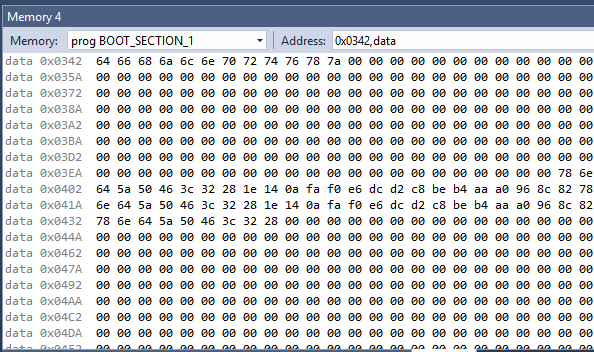
****

c) Memory spaces

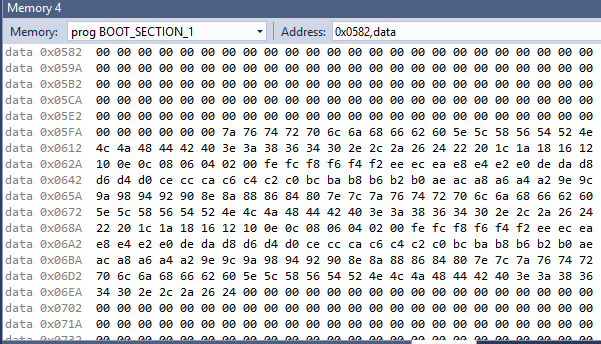
Memory space 0x222, first number placed is 0x24 (36) and last number is 0x7a (122)

****

Memory space 0x400, numbers divisible by 5

****

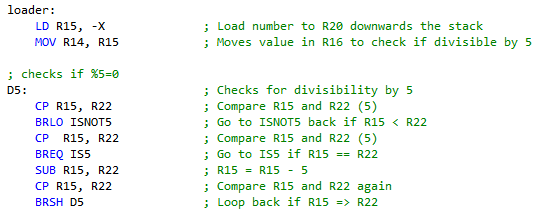
Memory space 0x600, numbers not divisible by 5

****

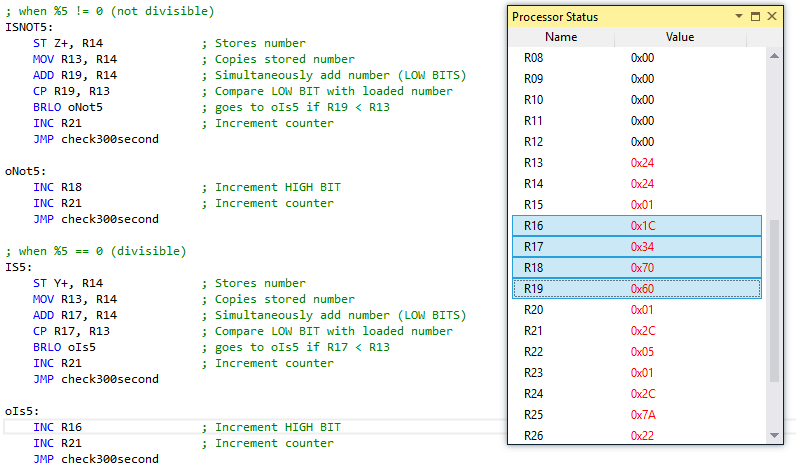
**Task 2/B**: Use X/Y/Z register addressing to parse through the 300 numbers, if the number is divisible by 5 store the number starting from memory location 0x0400, else store at location starting at 0x0600.

**Task 3/C**: Use X/Y/Z register addressing to simultaneously add numbers from memory location 0x0400 and 0x0600 and store the sums at R16:R17 and R18:R19 respectively. Do not worry about the overflow.

a) checking if each number is divisible by 5, goes to registers R16 and R17 if divisible by 5 and goes to registers R18 and R19 if not

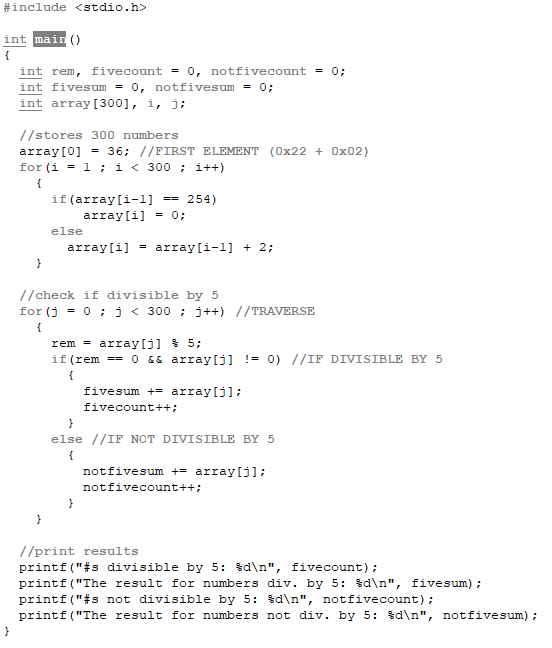


b) running the program, R16:R17 (divisible by 5) is 7220 (0x1C34) and R18:R19 (not divisible by 5) is 28768 (0x7060)

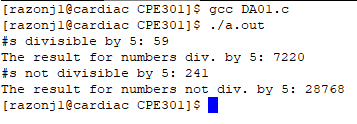


**Task 4/D**: Verify your algorithm and answers using C programming

a) The Code



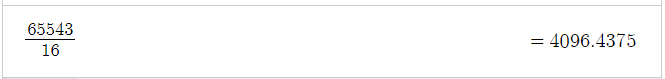
b) running the code, the outputs matches the hexadecimal results from Task 3/C, 7220 being 0x1C34 and 28768 being 0x7060



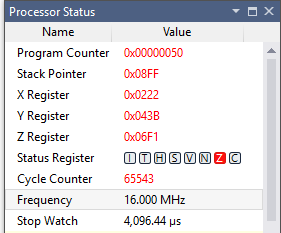
**Task 5/E**:Determine the execution time @ 16MHz/#cycles of your algorithm using the simulation.

Execution time (in microseconds) = # of cycles (65543 cycles) / frequency (16 MHz) = 4096.44 us

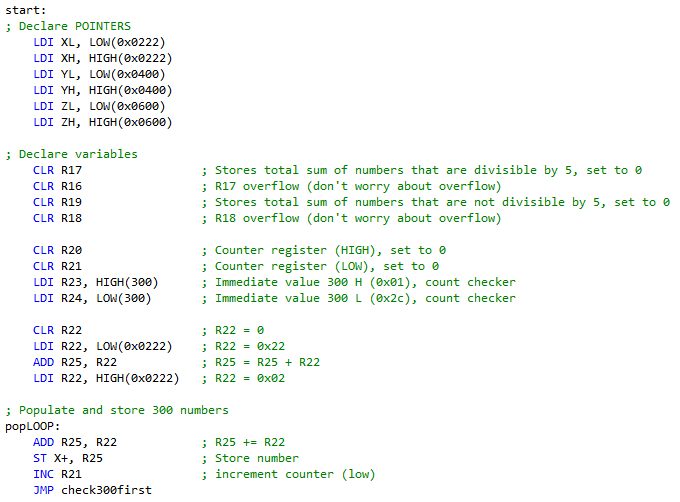
a) calculations on online scientific calculator

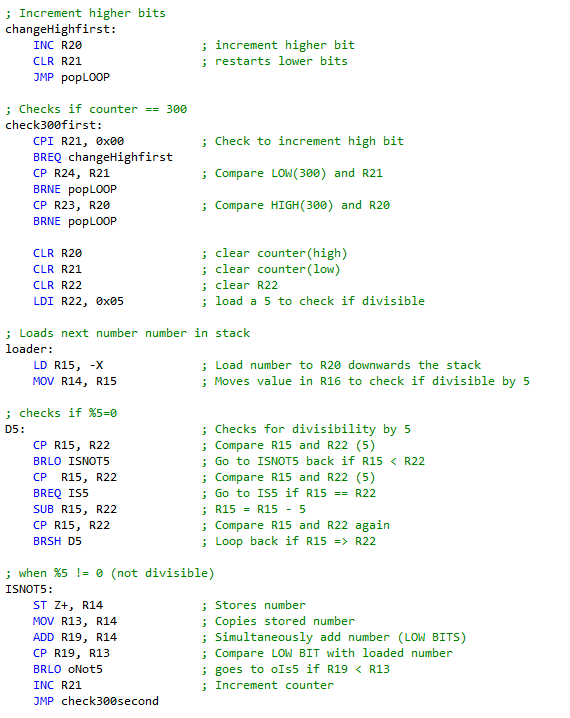


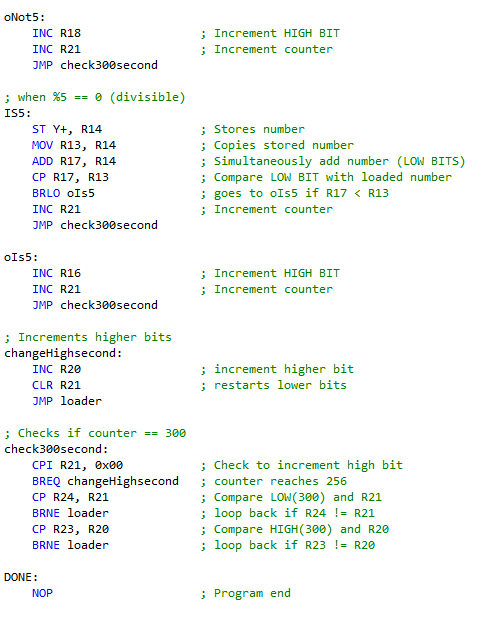
b) by changing the frequency on Atmel Studios, the stop watch’s execution time changed accordingly



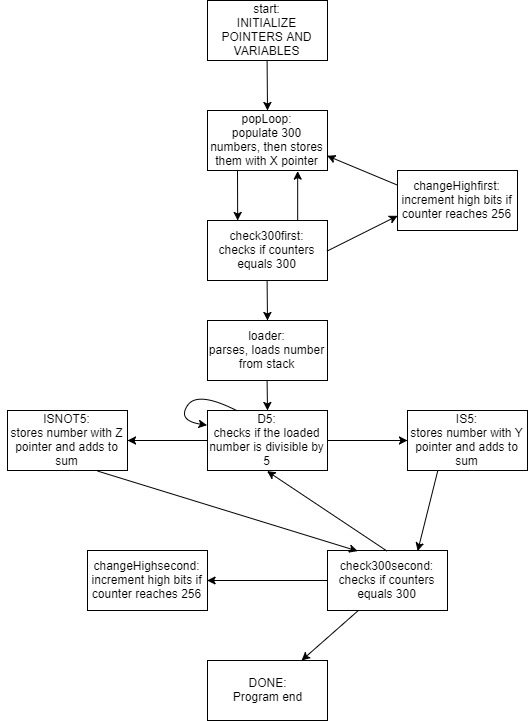
**FULL CODE**







**FLOW CHART**



**GITHUB LINK:** https://github.com/JeffinVegas/EmbSys.git

**Student Academic Misconduct Policy**

<http://studentconduct.unlv.edu/misconduct/policy.html>

“*This assignment submission is my own, original work*”.

Jeffrey Razon