CPE301 – SPRING 2019

Design Assignment 1B

Student Name: Tyler Gardenhire
Student #: 8000450294
Student Email: gardenhi@unlv.nevada.edu
Primary Github address: gardenhi@unlv.nevada.edu
Directory: https://github.com/tylergardenhire/submission_projects.git

Submit the following for all Labs:

1. In the document, for each task submit the modified or included code (only) with highlights and justifications of the modifications. Also, include the comments.
2. Use the previously create a Github repository with a random name (no CPE/301, Lastname, Firstname). Place all labs under the root folder ESD301/DA, sub-folder named LABXX, with one document and one video link file for each lab, place modified asm/c files named as LabXX-TYY.asm/c.
3. If multiple asm/c files or other libraries are used, create a folder LabXX-TYY and place these files inside the folder.
4. The folder should have a) Word document (see template), b) source code file(s) and other include files, c) text file with youtube video links (see template).

1. COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS

Atmel Studio 7 w/ AVR assembly and simulator used

2. INITIAL/MODIFIED/DEVELOPED CODE OF TASK 1/A

Assembly Code for AVR ATMEGA328p microcontroller:

```
;
; Project1B.asm
;
; Created: 2/23/2019 6:31:12 PM
; Author : Tyler Gardenhire
;


	.include <m328pdef.inc> ;standard library
	.EQU STARTADDS = 0x0200 ;starting address
	.EQU DIVBY3 = 0x0400    ;starting address of numbers divisible by 3
	.EQU NOTDIVBY3 = 0x0600 ;starting address of numbers not divisible by 3


	ldi r20, 99 ;counter
	ldi r21, 11 ;numbers stored in memory
	ldi r22, 0  ;zero

	ldi XL, low(STARTADDS)  ;stores lower address value (0x00)
	ldi XH, high(STARTADDS) ;stores higher address value (0x02)
	ldi YL, low(DIVBY3)     ;stores lower address value (0x00)
	ldi YH, high(DIVBY3)    ;stores higher address value (0x04)
	ldi ZL, low(NOTDIVBY3)  ;stores lower address value (0x00)
	ldi ZH, high(NOTDIVBY3) ;stores higher address value (0x06)

	store: st X+, r21          ;loop to store numbers
	inc r21                             ;increment the value to store
	dec r20                             ;decrement counter
	brne store                        ;if counter not equal to 0, loop to store

	ldi XL, low(STARTADDS)  ;load lower address value to read values from
	ldi XH, high(STARTADDS) ;load higher address value to read
	ldi r20, 99                       ;re-load counter

	check: ld r23, X+ ;checks for divisibility by 3
	mov r24, r23    ;use r24 as temp value for divisibility

	divisibility: subi r24, 3   ;subtract by 3
	breq divisibleby3                 ;if 0, value is divisible by 3
	brmi notdivisibleby3        ;if negative, value is not divisible by 3
	rjmp divisibility                 ;if neither, loop back to divisibility

	divisibleby3: st Y+, r23    ;store value in 0x0400 if divisible by 3
	add r16, r23                      ;add sum to r16
	adc r17, r22                      ;increase sum to 16 bits
	dec r20                                 ;decrement counter
	breq done                         ;if counter equals 0, done
	rjmp check                        ;if not, check next value
```

```
notdivisibleby3: st Z+, r23 ;store value in 0x0600
add r18, r23                ;add sum to r18
adc r19, r22                ;increase sum to 16 bits
dec r20                                 ;decrement counter
breq done                       ;if counter equals 0, done
rjmp check                      ;if not, check next value
```

3. DEVELOPED MODIFIED CODE OF TASK 2/A from TASK 1/A

C++ code to verify Assembly code:

```
#include <iostream>
using namespace std;

int main()
{
	int num = 11;
	int div3sum, notdiv3sum;

	for (int i = 0; i < 99; i++)
	{
		if ((num % 3) != 0)
			notdiv3sum += num;
		else
			div3sum += num;
		num++;
	}

	cout << "Sum of numbers divisible by 3 = " << hex << div3sum << endl;
	cout << "Sum of numbers not divisible by 3 = " << hex << notdiv3sum << endl;

return 0;
}
```

4. SCHEMATICS

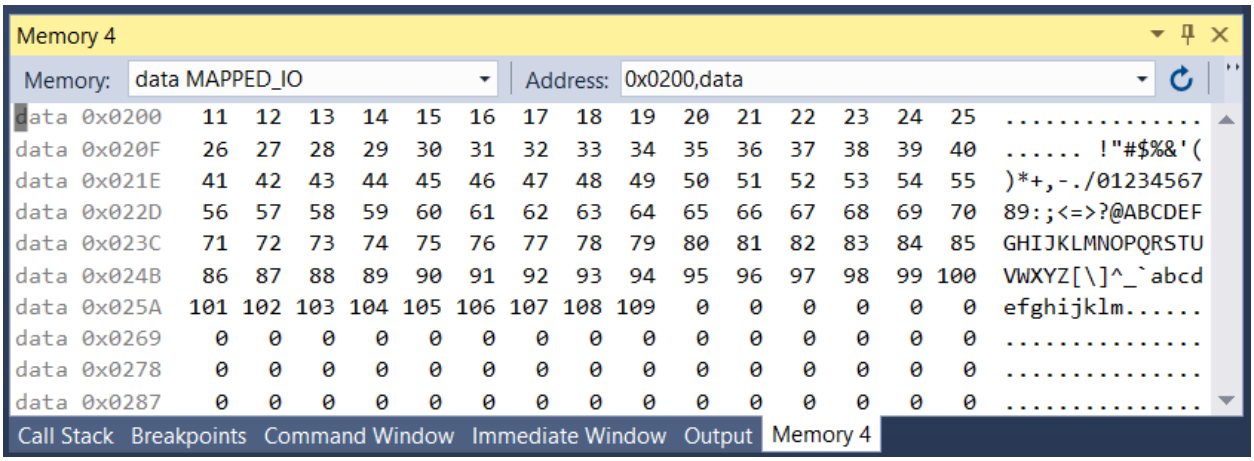
N/A

5. SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)

Task 1:

```
store: st X+, r21        ;loop to store numbers
inc r21                  ;increment the value to store
dec r20                  ;decrement counter
brne store               ;if counter not equal to 0, loop to store
```

This portion of code uses the X pointer to store all 99 numbers.



Starting address (STARTADDS) is 0x0200, all 99 numbers stored in consecutive order at this address.

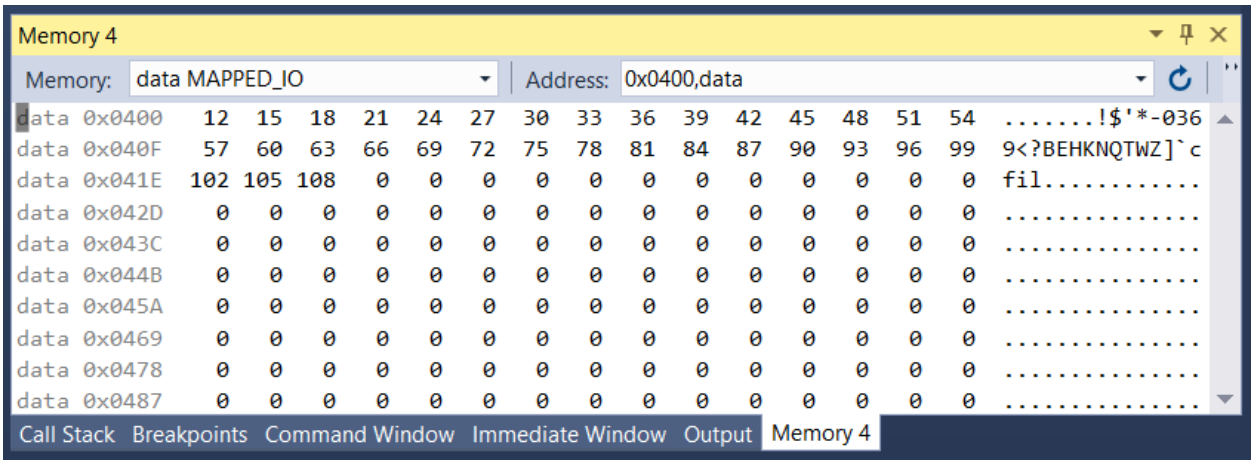
Task 2:

```
divisibility: subi r24, 3    ;subtract by 3
breq divisibleby3            ;if 0, value is divisible by 3
brmi notdivisibleby3         ;if negative, value is not divisible by 3
rjmp divisibility            ;if neither, loop back to divisibility
```

This portion of code determines whether the number is divisible by 3 or not by looping a subtract instruction. If the value ends up being 0, the number is divisible by 3 but if the value ends up being negative, the number is not divisible by 3.

```
divisibleby3: st Y+, r23     ;store value in 0x0400 if divisible by 3
add r16, r23                 ;add sum to r16
adc r17, r22                 ;increase sum to 16 bits
dec r20                      ;decrement counter
breq done                    ;if counter equals 0, done
rjmp check                   ;if not, check next value
```

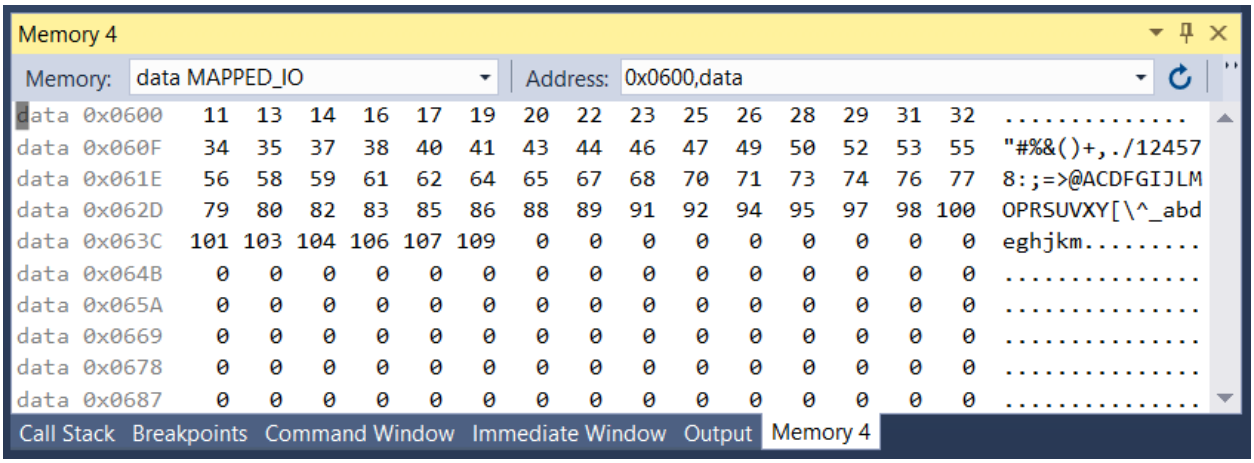
This portion of code stores all numbers divisible by 3 into the address located in the Y pointer (0x0400) and stores their sum into registers r16 and r17.



All numbers divisible by 3 are in order, starting at address 0x0400.

```
notdivisibleby3: st Z+, r23 ;store value in 0x0600
add r18, r23                ;add sum to r18
adc r19, r22                ;increase sum to 16 bits
dec r20                     ;decrement counter
breq done                   ;if counter equals 0, done
rjmp check                  ;if not, check next value
```

This portion of code stores all numbers not divisible by 3 into the address located in the Z pointer (0x0600) and stores their sum into registers r18 and r19.



All numbers not divisible by 3 are in order, starting at address 0x0600.

Task 3:

R16	0xBC
R17	0x07
R18	0x78
R19	0x0F

The screenshot above shows registers r16-r17 filled with the sum of numbers divisible by 3 (0x07BC) and registers r18-r19 filled with the sum of numbers not divisible by 3 (0x0F78).

Task 4:

```cpp
#include <iostream>
using namespace std;

int main()
{
	int num = 11;
	int div3sum, notdiv3sum;

	for (int i = 0; i < 99; i++)
	{
		if ((num % 3) != 0)
			notdiv3sum += num;
		else
			div3sum += num;
		num++;
	}

	cout << "Sum of numbers divisible by 3 = " << hex << div3sum << endl;
	cout << "Sum of numbers not divisible by 3 = " << hex << notdiv3sum << endl;

return 0;
```

This C++ code verifies the assembly code by using for loops and modulus to check for numbers divisible by 3.

```
Sum of numbers divisible by 3 = 7bc
Sum of numbers not divisible by 3 = f78
```

The output of the C++ code verifies our above values (0x07BC and 0x0F78).

Task 5:

Cycle Counter	11625
Frequency	16.000 MHz
Stop Watch	726.56 µs

The above screenshot shows the execution time and cycle counter when using a frequency of 16 MHz.

6. SCREENSHOT OF EACH DEMO (BOARD SETUP)

N/A

7. VIDEO LINKS OF EACH DEMO

N/A

8. GITHUB LINK OF THIS DA

https://github.com/tylergardenhire/submission_projects.git

Student Academic Misconduct Policy
http://studentconduct.unlv.edu/misconduct/policy.html

"*This assignment submission is my own, original work*".
TYLER GARDENHIRE