Bryan Takemoto

CPE301 – SPRING 2018

Design Assignment 1

**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 | SUM OF STARTADDS CODE [TASK 1] |  |  |
| 2. | CHECK DIVISIBILITY CODE [TASK 2] |  |  |
| 3. | SUM OF NUMBERS THAT ARE/AREN’T DIVISIBLE CODE [TASK 3] |  |  |
| 4. | C CODE VERIFICATION OF ALGORITHM [TASK 4] |  |  |
| 5. | EXECUTION TIME [TASK 5] |  |  |
| 6. | COMPLETE MAIN ASSEMBLY CODE |  |  |
| 7. | SCREENSHOTS OF THE OUTPUTS |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. **SUM OF STARTADDS CODE [TASK 1]**

; Task #1

; Initializes the pointer 0x0222 of register X

LDI XL, LOW(STARTADDS)

LDI XH, HIGH(STARTADDS)

LDI R17, 3 ; Outer loop which counts to 3 (R17)

LP0:

LDI R16, 100 ; Inner loop which counts to 100 (R16)

LP1:

MOV R0, XL ; Use R0 for LOW(STARTADDS)

MOV R1, XH ; Use R1 for HIGH(STARTADDS)

ADD R0, R1 ; LOW(STARTADDS) + HIGH(STARTADDS)

ST X+, R0 ; Stores the results

DEC R16 ; Updates inner loop counter

BRNE LP1 ; Checks inner loop condition (R16 != 0)

DEC R17 ; Updates outer loop counter

BRNE LP0 ; Checks outer loop condition (R17 != 0)

1. **CHECK DIVISIBILITY CODE [TASK 2]**

; Task #2

; Initializes the pointer 0x0222 of register X

LDI XL, LOW(STARTADDS)

LDI XH, HIGH(STARTADDS)

; Initializes the pointer 0x0400 of register Y

LDI YL, LOW(MEM\_DIV\_5)

LDI YH, HIGH(MEM\_DIV\_5)

; Initializes the pointer 0x0600 of register Z

LDI ZL, LOW(MEM\_NDIV\_5)

LDI ZH, HIGH(MEM\_NDIV\_5)

LDI R24, 0x00 ; Count the number of numbers divisible by 5

LDI R25, 0x00 ; Count the number of numbers isn't divisible by 5

LDI R17, 3 ; Outer loop which counts to 3 (R17)

LP2:

LDI R16, 100 ; Inner loop which counts to 100 ($16)

LP3:

LD R18, X ; Loads the number to be modified (masking) into R18

LD R19, X ; Saves the unmodified number into R19

SUB\_LOOP:

CPI R18, 0x05 ; Checks if the current result is divisible by 5

BREQ DIV\_5

SUBI R18, 0x05 ; Subtracts the number being tested by 5

BRLO NOT\_DIV\_5 ; If an underflow occurs, the number is not divisible by 5

RJMP SUB\_LOOP

DIV\_5:

ST Y+, R19 ; Stores the number that was divisible by 5

INC R24 ; Updates count (Divisible)

RJMP END\_DIV

NOT\_DIV\_5:

ST Z+, R19 ; Stores the number that was not divisible by 5

INC R25 ; Updates count (Not Divisible)

END\_DIV:

DEC R16 ; Updates inner loop counter

BRNE LP3 ; Checks inner loop condition (R16 != 0)

DEC R17 ; Updates outer loop counter

BRNE LP2 ; Checks outer loop condition (R17 != 0)

1. **SUM OF NUMBERS THAT ARE/AREN’T DIVISIBILE BY 5 CODE [TASK 3]**

; Task #3

; Initializes the pointer registers X and Y

LDI XL, LOW(MEM\_DIV\_5); Pointer to the array of numbers that are divisible by 5

LDI XH, HIGH(MEM\_DIV\_5)

LDI YL, LOW(MEM\_NDIV\_5) ; Pointer to the array of numbers that aren't divisible by 5

LDI YH, HIGH(MEM\_NDIV\_5)

; Clear the registers

CLR R16

CLR R17

CLR R18

CLR R19

CLR R22 ; Zeroed upper 8-bit for the 16-bit operands

SUM\_0:

LD R20, X+ ; Grabs the number from the array and updates the pointer

ADD R16, R20 ; Adds lower 8-bits

ADC R17, R22 ; Adds upper 8-bits

DEC R24 ; Update counter

BRNE SUM\_0

SUM\_1:

LD R21, Y+ ; Grabs the number form the array and updates the pointer

ADD R18, R21 ; Adds lower 8-bits

ADC R19, R22 ; Adds upper 8-bits

DEC R25 ; Update counter

BRNE SUM\_1

END:

RJMP END

1. **C CODE VERIFICATION OF ALGORITHM [TASK 4]**

#include <stdio.h>

#include <stdlib.h>

int main()

{

int arr\_0[30][10];

int arr\_1[30][10];

int arr\_2[30][10];

int \*divArr;

int \*ndivArr;

int div\_arr[300];

int ndiv\_arr[300];

int a = 0x0222;

int b, c;

int sum;

int i, j;

int temp;

int div\_count = 0;

int ndiv\_count = 0;

printf("\nTask #1:\n");

printf("Summation Array\n");

for(i = 0; i < 30; i++)

{

for(j = 0; j < 10; j++)

{

b = (a & 0x0000FF00) >> 8;

c = a & 0x000000FF;

sum = b + c;

sum = sum & 0x000000FF;

arr\_0[i][j] = sum;

printf("%3x", arr\_0[i][j]);

a++;

}

printf("\n");

}

printf("\nTask #2:\n");

for(i = 0; i < 30; i++)

{

for(j = 0; j < 10; j++)

{

temp = arr\_0[i][j] % 0x05;

if(arr\_0[i][j] == 0 || temp != 0)

{

arr\_2[i][j] = arr\_0[i][j];

arr\_1[i][j] = 0x0;

ndiv\_count++;

}

else

{

arr\_1[i][j] = arr\_0[i][j];

arr\_2[i][j] = 0x0;

div\_count++;

}

}

}

divArr = malloc(div\_count\*sizeof(int));

temp = 0;

for(i = 0; i < 30; i++)

{

for(j = 0; j < 10; j++)

{

if(arr\_1[i][j] > 0)

{

divArr[temp] = arr\_1[i][j];

temp++;

}

}

}

ndivArr = malloc(ndiv\_count\*sizeof(int));

temp = 0;

for(i = 0; i < 30; i++)

{

for(j = 0; j < 10; j++)

{

if(arr\_2[i][j] > 0)

{

ndivArr[temp] = arr\_2[i][j];

temp++;

}

}

}

printf("\nNumbers divisible by 5\n");

temp = 0;

for(i = 0; i < div\_count; i++)

{

printf("%3x", divArr[i]);

temp++;

if(temp == 10)

{

printf("\n");

temp = 0;

}

}

printf("\nNumbers aren't divisible by 5\n");

temp = 0;

for(i = 0; i < ndiv\_count; i++)

{

printf("%3x", ndivArr[i]);

temp++;

if(temp == 10)

{

printf("\n");

temp = 0;

}

}

printf("\nTotal numbers that are divisible by 5: %d", div\_count);

printf("\nTotal numbers that aren't divisible by 5: %d\n", ndiv\_count);

printf("\nTask #3:\n");

sum = 0;

for(i = 0; i < div\_count; i++)

{

sum += divArr[i];

}

printf("Sum of Div: %x\n", sum);

sum = 0;

for(i = 0; i < ndiv\_count; i++)

{

sum += ndivArr[i];

}

printf("Sum of not Div: %x\n", sum);

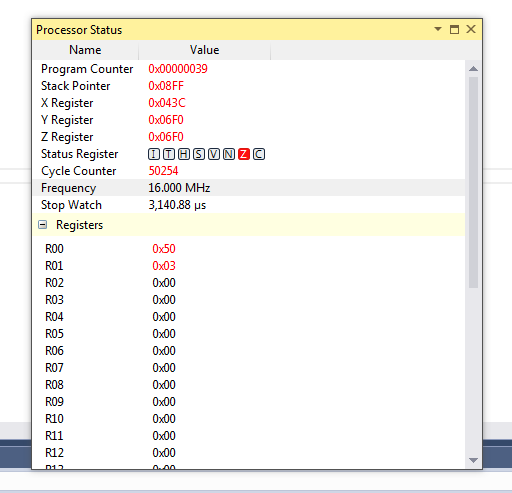
free(divArr);

free(ndivArr);

return 0;

}

1. **EXECUTION TIME [TASK 5]**



1. **COMPLETE MAIN ASSEMBLY CODE**

; Addresses in SRAM to store an array of numbers

.EQU STARTADDS = 0x0222

.EQU MEM\_DIV\_5 = 0x0400

.EQU MEM\_NDIV\_5 = 0x0600

.ORG 0

; Task #1

; Initializes the pointer 0x0222 of register X

LDI XL, LOW(STARTADDS)

LDI XH, HIGH(STARTADDS)

LDI R17, 3 ; Outer loop which counts to 3 (R17)

LP0:

LDI R16, 100 ; Inner loop which counts to 100 (R16)

LP1:

MOV R0, XL ; Use R0 for LOW(STARTADDS)

MOV R1, XH ; Use R1 for HIGH(STARTADDS)

ADD R0, R1 ; LOW(STARTADDS) + HIGH(STARTADDS)

ST X+, R0 ; Stores the results

DEC R16 ; Updates inner loop counter

BRNE LP1 ; Checks inner loop condition (R16 != 0)

DEC R17 ; Updates outer loop counter

BRNE LP0 ; Checks outer loop condition (R17 != 0)

; Task #2

; Initializes the pointer 0x0222 of register X

LDI XL, LOW(STARTADDS)

LDI XH, HIGH(STARTADDS)

; Initializes the pointer 0x0400 of register Y

LDI YL, LOW(MEM\_DIV\_5)

LDI YH, HIGH(MEM\_DIV\_5)

; Initializes the pointer 0x0600 of register Z

LDI ZL, LOW(MEM\_NDIV\_5)

LDI ZH, HIGH(MEM\_NDIV\_5)

LDI R24, 0x00 ; Count the number of numbers divisible by 5

LDI R25, 0x00 ; Count the number of numbers isn't divisible by 5

LDI R17, 3 ; Outer loop which counts to 3 (R17)

LP2:

LDI R16, 100 ; Inner loop which counts to 100 ($16)

LP3:

LD R18, X ; Loads the number to be modified (masking) into R18

LD R19, X+ ; Saves the unmodified number into R19

SUB\_LOOP:

CPI R18, 0x05 ; Checks if the current result is divisible by 5

BREQ DIV\_5

SUBI R18, 0x05 ; Subtracts the number being tested by 5

BRLO NOT\_DIV\_5 ; If an underflow occurs, the number is not divisible by 5

RJMP SUB\_LOOP

DIV\_5:

ST Y+, R19 ; Stores the number that was divisible by 5

INC R24 ; Updates count (Divisible)

RJMP END\_DIV

NOT\_DIV\_5:

ST Z+, R19 ; Stores the number that was not divisible by 5

INC R25 ; Updates count (Not Divisible)

END\_DIV:

DEC R16 ; Updates inner loop counter

BRNE LP3 ; Checks inner loop condition (R16 != 0)

DEC R17 ; Updates outer loop counter

BRNE LP2 ; Checks outer loop condition (R17 != 0)

; Task #3

; Initializes the pointer registers X and Y

LDI XL, LOW(MEM\_DIV\_5) ; Pointer to the array of numbers that are divisible by 5

LDI XH, HIGH(MEM\_DIV\_5)

LDI YL, LOW(MEM\_NDIV\_5) ; Pointer to the array of numbers that aren't divisible by 5

LDI YH, HIGH(MEM\_NDIV\_5)

; Clear the registers

CLR R16

CLR R17

CLR R18

CLR R19

CLR R22 ; Zeroed upper 8-bit for the 16-bit operands

SUM\_0:

LD R20, X+ ; Grabs the number from the array and updates the pointer

ADD R16, R20 ; Adds lower 8-bits

ADC R17, R22 ; Adds upper 8-bits

DEC R24 ; Update counter

BRNE SUM\_0

SUM\_1:

LD R21, Y+ ; Grabs the number form the array and updates the pointer

ADD R18, R21 ; Adds lower 8-bits

ADC R19, R22 ; Adds upper 8-bits

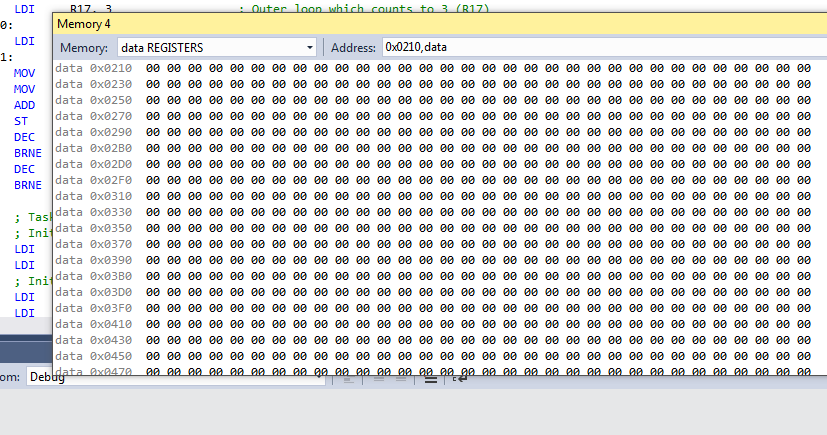
DEC R25 ; Update counter

BRNE SUM\_1

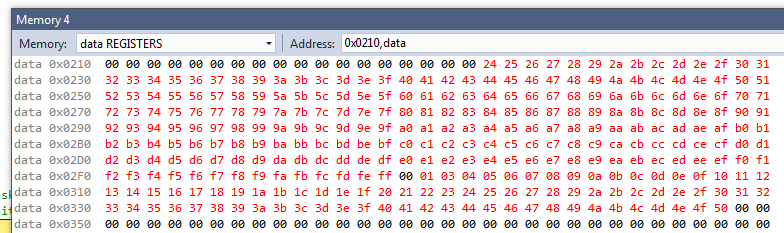
END:

RJMP END

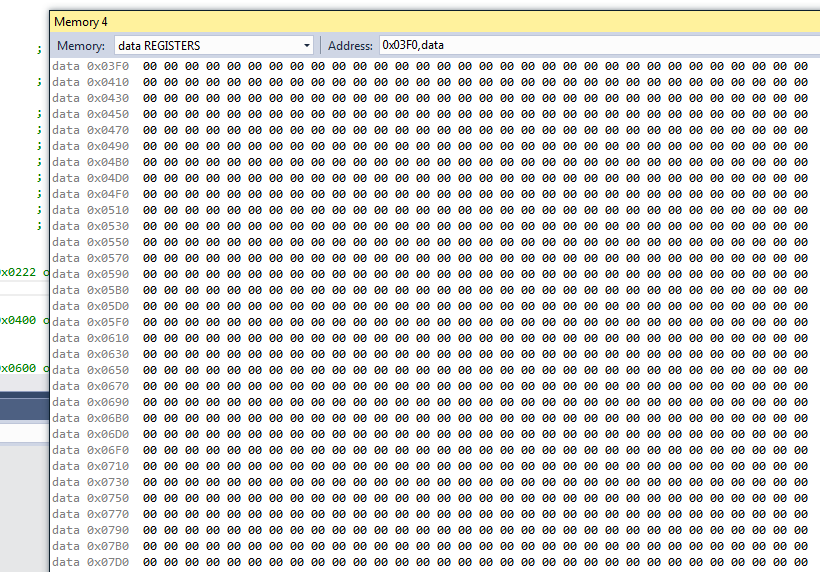
1. **SCREENSHOTS OF THE OUTPUTS**

****

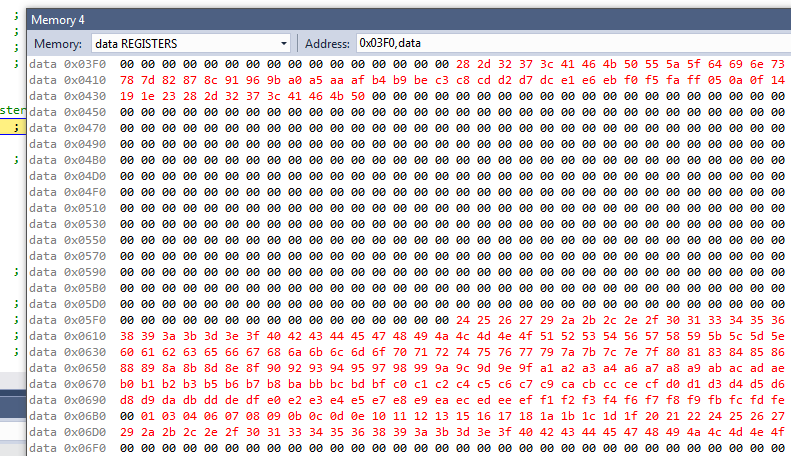
***Figure 1: Before storing [Task 1]***



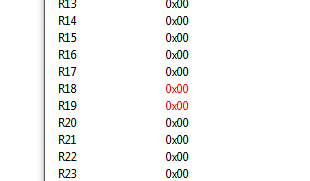
***Figure 2: After storing [Task 1]***



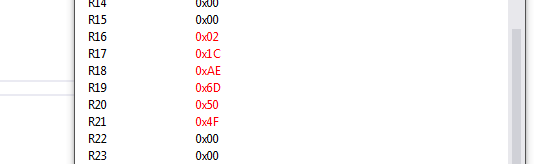
***Figure 3: Before storing [Task 2]***



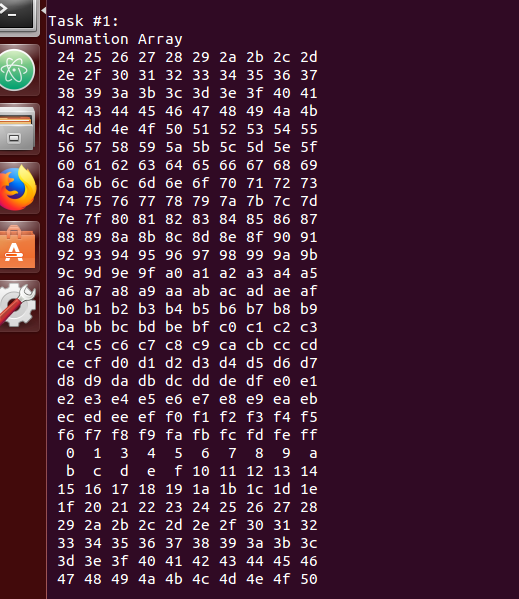
***Figure 4: After storing [Task 2]***



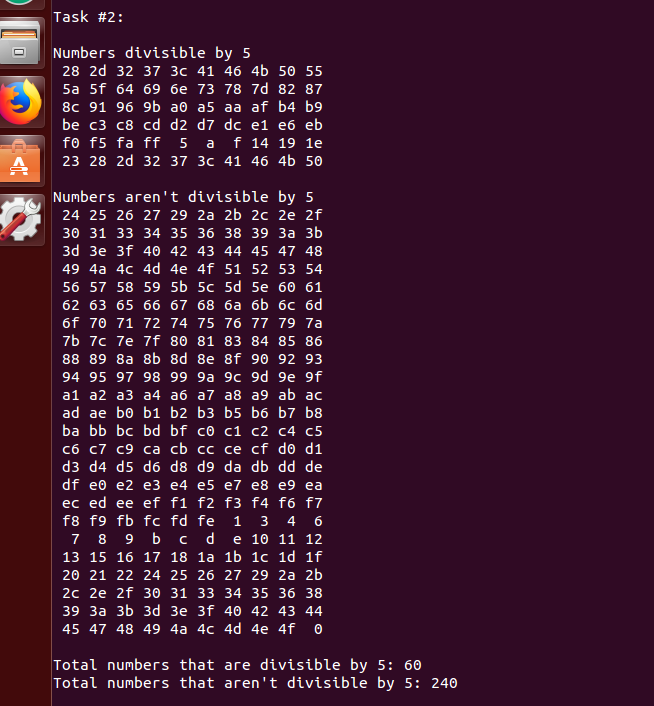
***Figure 5: Before summation [Task 3]***



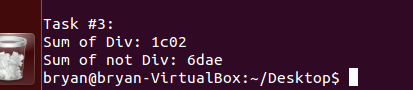
***Figure 6: After summation [Task 3]***

******

***Figure 7: C Verification of Task 1 [Task 4]***

******

***Figure 8: C Verification of Task 2 [Task 4]***

******

***Figure 8: C Verification of Task 3 [Task 4]***

**Student Academic Misconduct Policy**

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

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

BRYAN TAKEMOTO