

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 STARTADD=0x0222 location. Populate the value of the memory location by adding high(STARTADD) and low(STARTADD) . Use the X/ Y/Z registers as pointers to fill up 300 numbers.

a) Declaring X/Y/Z registers as pointers

```
start:
; Declare POINTERS
LDI XL, LOW(0x0222)
LDI XH, HIGH(0x0222)
LDI YL, LOW(0x0400)
LDI YH, HIGH(0x0400)
LDI ZL, LOW(0x0600)
LDI ZH, HIGH(0x0600)
```

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

```
CLR R22          ; R22 = 0
LDI R22, LOW(0x0222) ; R22 = 0x22
ADD R25, R22      ; R25 = R25 + R22
LDI R22, HIGH(0x0222) ; R22 = 0x02

; Populate and store 300 numbers
popLOOP:
ADD R25, R22      ; R25 += R22
ST X+, R25        ; Store number
INC R21           ; increment counter (low)
JMP check300first

; Increment higher bits
changeHighfirst:
INC R20           ; increment higher bit
CLR R21           ; restarts lower bits
JMP popLOOP

; Checks if counter == 300
check300first:
CPI R21, 0x00     ; Check to increment high bit
BREQ changeHighfirst
CP R24, R21        ; Compare LOW(300) and R21
BRNE popLOOP
CP R23, R20        ; Compare HIGH(300) and R20
BRNE popLOOP
```

**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

```

loader:
    LD R15, -X          ; Load number to R20 downwards the stack
    MOV R14, R15        ; Moves value in R16 to check if divisible by 5

; checks if %5=0
D5:
    CP R15, R22          ; Compare R15 and R22 (5)
    BRLO ISNOT5          ; Go to ISNOT5 back if R15 < R22
    CP R15, R22          ; Compare R15 and R22 (5)
    BREQ IS5            ; Go to IS5 if R15 == R22
    SUB R15, R22         ; R15 = R15 - 5
    CP R15, R22          ; Compare R15 and R22 again
    BRSH D5             ; Loop back if R15 >= R22

```

b) running the program, R17 (divisible by 5) is 0x34 (52) and R19 (not divisible by 5) is 0x60 (96), the total sum of numbers divisible by 5 is 7220 (0x1C34) and the total sum of numbers not divisible by 5 is 28768 (0x7060)

```

; when %5 != 0 (not divisible)
ISNOT5:
    ST Z+, R14          ; Stores number
    MOV R13, R14        ; Copies stored number
    ADD R19, R14        ; Simultaneously add number (LOW BITS)
    CP R19, R13          ; Compare LOW BIT with loaded number
    BRLO oNot5          ; goes to oIs5 if R19 < R13
    INC R21             ; Increment counter
    JMP check300second

oNot5:
    INC R18             ; Increment HIGH BIT
    INC R21             ; Increment counter
    JMP check300second

; when %5 == 0 (divisible)
IS5:
    ST Y+, R14          ; Stores number
    MOV R13, R14        ; Copies stored number
    ADD R17, R14        ; Simultaneously add number (LOW BITS)
    CP R17, R13          ; Compare LOW BIT with loaded number
    BRLO oIs5          ; goes to oIs5 if R17 < R13
    INC R21             ; Increment counter
    JMP check300second

oIs5:
    INC R16             ; Increment HIGH BIT
    INC R21             ; Increment counter
    JMP check300second

```

Name	Value
R08	0x00
R09	0x00
R10	0x00
R11	0x00
R12	0x00
R13	0x24
R14	0x24
R15	0x01
R16	0x1C
R17	0x34
R18	0x70
R19	0x60
R20	0x01
R21	0x2C
R22	0x05
R23	0x01
R24	0x2C
R25	0x7A
R26	0x22

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

##### a) The Code

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

int main()
{
    int rem, fivecount = 0, notfivecount = 0;
    int fivesum = 0, notfivesum = 0;
    int array[300];

    //stores 300 numbers
    array[0] = 36; //FIRST ELEMENT (0x22 + 0x02)
    for(int i = 1 ; i < 300 ; i++)
    {
        if(array[i-1] == 254)
            array[i] = 0;
        else
            array[i] = array[i-1] + 2;
    }

    //check if divisible by 5
    for(int j = 0 ; j < 300 ; j++) //TRAVERSE
    {
        rem = array[j] % 5;
        if(rem == 0 && array[j] != 0) //IF DIVISIBLE BY 5
        {
            fivesum += array[j];
            fivecount++;
        }
        else //IF NOT DIVISIBLE BY 5
        {
            notfivesum += array[j];
            notfivecount++;
        }
    }

    //print results
    cout << "#s divisible by 5: " << fivecount << endl;
    cout << "The result for numbers div. by 5: ";
    //cout << fivesum%256 << endl;
    cout << fivesum << endl;

    cout << "#s not divisible by 5: " << notfivecount << endl;
    cout << "The result for numbers not div. by 5: ";
    //cout << notfivesum%256 << endl;
    cout << notfivesum << endl;
}
```

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

```
[razonjl@bobby CPE301]$ ./a.out
#s divisible by 5: 59
The result for numbers div. by 5: 7220
#s not divisible by 5: 241
The result for numbers not div. by 5: 28768
[razonjl@bobby CPE301]$
```

**Task 5/E:** Determine the execution time @ 16MHz/#cycles of your algorithm using the simulation.  
 Execution time (in microseconds) = # of cycles (64663 cycles) / frequency (16 MHz) = 4041.44

a) calculations on online scientific calculator

$$\frac{65543}{16} = 4096.4375$$

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

Processor Status	
Name	Value
Program Counter	0x00000050
Stack Pointer	0x08FF
X Register	0x0222
Y Register	0x043B
Z Register	0x06F1
Status Register	I T H S V N Z C
Cycle Counter	65543
Frequency	16.000 MHz
Stop Watch	4,096.44 µs

## FULL CODE

```
start:
; Declare POINTERS
LDI XL, LOW(0x0222)
LDI XH, HIGH(0x0222)
LDI YL, LOW(0x0400)
LDI YH, HIGH(0x0400)
LDI ZL, LOW(0x0600)
LDI ZH, HIGH(0x0600)

; Declare variables
CLR R17          ; Stores total sum of numbers that are divisible by 5, set to 0
CLR R16          ; R17 overflow (don't worry about overflow)
CLR R19          ; Stores total sum of numbers that are not divisible by 5, set to 0
CLR R18          ; R18 overflow (don't worry about overflow)

CLR R20          ; Counter register (HIGH), set to 0
CLR R21          ; Counter register (LOW), set to 0
LDI R23, HIGH(300) ; Immediate value 300 H (0x01), count checker
LDI R24, LOW(300)  ; Immediate value 300 L (0x2c), count checker

CLR R22          ; R22 = 0
LDI R22, LOW(0x0222) ; R22 = 0x22
ADD R25, R22      ; R25 = R25 + R22
LDI R22, HIGH(0x0222) ; R22 = 0x02

; Populate and store 300 numbers
popLOOP:
ADD R25, R22      ; R25 += R22
ST X+, R25        ; Store number
INC R21           ; increment counter (low)
JMP check300first
```

```

; Increment higher bits
changeHighfirst:
    INC R20          ; increment higher bit
    CLR R21          ; restarts lower bits
    JMP popLOOP

; Checks if counter == 300
check300first:
    CPI R21, 0x00    ; Check to increment high bit
    BREQ changeHighfirst
    CP R24, R21       ; Compare LOW(300) and R21
    BRNE popLOOP
    CP R23, R20       ; Compare HIGH(300) and R20
    BRNE popLOOP

    CLR R20          ; clear counter(high)
    CLR R21          ; clear counter(low)
    CLR R22          ; clear R22
    LDI R22, 0x05    ; load a 5 to check if divisible

; Loads next number number in stack
loader:
    LD R15, -X        ; Load number to R20 downwards the stack
    MOV R14, R15      ; Moves value in R16 to check if divisible by 5

; checks if %5=0
D5:
    CP R15, R22       ; Compare R15 and R22 (5)
    BRLO ISNOT5       ; Go to ISNOT5 back if R15 < R22
    CP R15, R22       ; Compare R15 and R22 (5)
    BREQ IS5         ; Go to IS5 if R15 == R22
    SUB R15, R22      ; R15 = R15 - 5
    CP R15, R22       ; Compare R15 and R22 again
    BRSH D5          ; Loop back if R15 >= R22

; when %5 != 0 (not divisible)
ISNOT5:
    ST Z+, R14        ; Stores number
    MOV R13, R14      ; Copies stored number
    ADD R19, R14      ; Simultaneously add number (LOW BITS)
    CP R19, R13       ; Compare LOW BIT with loaded number
    BRLO oNot5        ; goes to oIs5 if R19 < R13
    INC R21           ; Increment counter
    JMP check300second

oNot5:
    INC R18           ; Increment HIGH BIT
    INC R21           ; Increment counter
    JMP check300second

; when %5 == 0 (divisible)
IS5:
    ST Y+, R14        ; Stores number
    MOV R13, R14      ; Copies stored number
    ADD R17, R14      ; Simultaneously add number (LOW BITS)
    CP R17, R13       ; Compare LOW BIT with loaded number
    BRLO oIs5        ; goes to oIs5 if R17 < R13
    INC R21           ; Increment counter
    JMP check300second

oIs5:
    INC R16           ; Increment HIGH BIT
    INC R21           ; Increment counter
    JMP check300second

; Increments higher bits
changeHighsecond:
    INC R20          ; increment higher bit
    CLR R21          ; restarts lower bits
    JMP loader

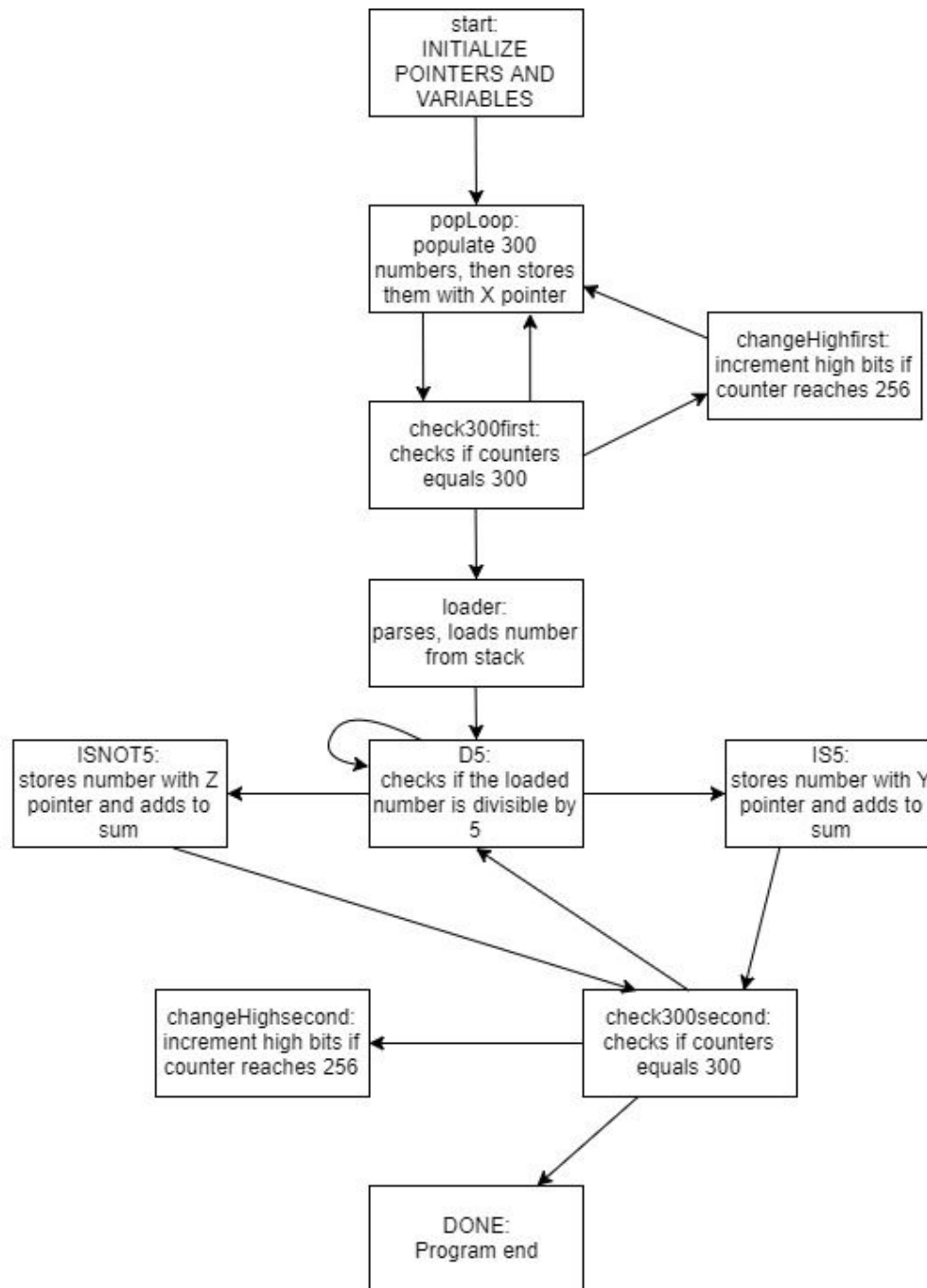
; Checks if counter == 300
check300second:
    CPI R21, 0x00    ; Check to increment high bit
    BREQ changeHighsecond ; counter reaches 256
    CP R24, R21       ; Compare LOW(300) and R21
    BRNE loader      ; loop back if R24 != R21
    CP R23, R20       ; Compare HIGH(300) and R20
    BRNE loader      ; loop back if R23 != R20

DONE:
    NOP              ; Program end

```



## 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