Dominique Anguiano

CPE301 – SPRING 2016

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. | INITIAL CODE OF TASK 1/A |  |  |
| 2. | SCREENSHOTS OF EACH TASK OUTPUT |  |  |
| 3. | FLOW CHART OF THE ASSEMBLY CODE |  |  |
| 4. | GITHUB LINK OF THE DA |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 1. | INITIAL CODE OF TASK 1/A |  |  |

;

; DA 1 Task 1.asm

;

; Created: 2/21/2016 8:05:21 PM

; Author : Dominique

;

.macro initstack ; Initialize Stack

ldi r16, high(RAMEND)

out sph, r16

ldi r16, low(RAMEND)

out spl, r16

.endmacro

start:

initstack

ldi zh, high(RAMEND) ; Z register will hold the value for

ldi zl, low(RAMEND) ; Ram\_middle

mov r16, zh ; Copy contents of upper 8 bits to r16

andi r16, 0x01 ; Obtain the LSB of the upper 8 bits

lsr zh ; Shift z registers right to divide by 2

lsr zl

cpi r16, 0x01 ; Check if lsb or upper 8 bits is 1

breq setOne ; Branch if equal

jmp dataStore

setOne:

ori zl, 0x80 ; Set msb of lower 8 bits to the same value

; as the lsb of the upper 8 bits

dataStore:

ldi r16, 25 ; Load 0 into r16

dataStoreLoop: ; Loop to store 25 values

dec r16 ; Increment r16

mov r17, zl ; Copy lower 8 bits into r17

st z+, r17 ; Store value of r17 into address z

cpi r16, 0 ; See if r16 is equal to 0

brne dataStoreLoop ; Loop if not equal

ldi r16, 25 ; set counter to 25

ldi r20, 0 ; Running sum for values divisible by 7

ldi r21, 0

ldi r23, 0 ; Running sum for values divisible by 3

ldi r24, 0

divisible:

ld r17, -z ; Load the value stored at address z

mov r18, r17 ; Copy the value into r18

div7:

subi r18, 7 ; Subtract 7 from r18

brcs div3 ; Branch if less than 0 to div3

cpi r18, 0 ; Compare r18 to 0

breq add7 ; if the values are equal jump to add7 label

jmp div7 ; loop again

add7:

add r20, r17 ; add the read value to r20

brcs carryAdd7 ; branch if a carry is set

jmp div3 ; branch to the division by 3 check

carryAdd7:

inc r21 ; increment the high bits of the running sum

clc ; clear the carry flag

jmp div3 ; branch to the division by 3 check

div3:

clc ; clear carry

mov r18, r17 ; copy value from r17 to r18

div3loop:

subi r18, 3 ; Subtract 3 from r18

brcs checkDone ; Branch if less than 0 for the subtraction

cpi r18, 0 ; Compare the value r17 to 0

breq add3 ; Branch if the value is 0 to add3 label

jmp div3loop

add3:

add r23, r17 ; add value in r17 to the running sum

brcs carryAdd3 ; branch if carry is set

jmp checkDone ; Branch to the checkDone flag

carryAdd3:

inc r24 ; Increment the high bits of the running sum

clc ; Clear the carry flag

jmp checkDone ; jump to checkDone flag

checkDone:

clc

dec r16 ; Decrement counter stored in r16

cpi r16, 0 ; Loop again if the counter is not equal to 0

brne divisible ; Branch to divisible label

cpi r21, 0 ; Check if the upper bits are greater than 0

brne notZero ; If the upper bits are greater than 0, jump

cpi r24, 0 ; If the upper bits are greater than 0, jump

brne notZero

jmp done ; Jump to the done label

notZero:

ldi r16, 8 ; Load 8 into the r16 register

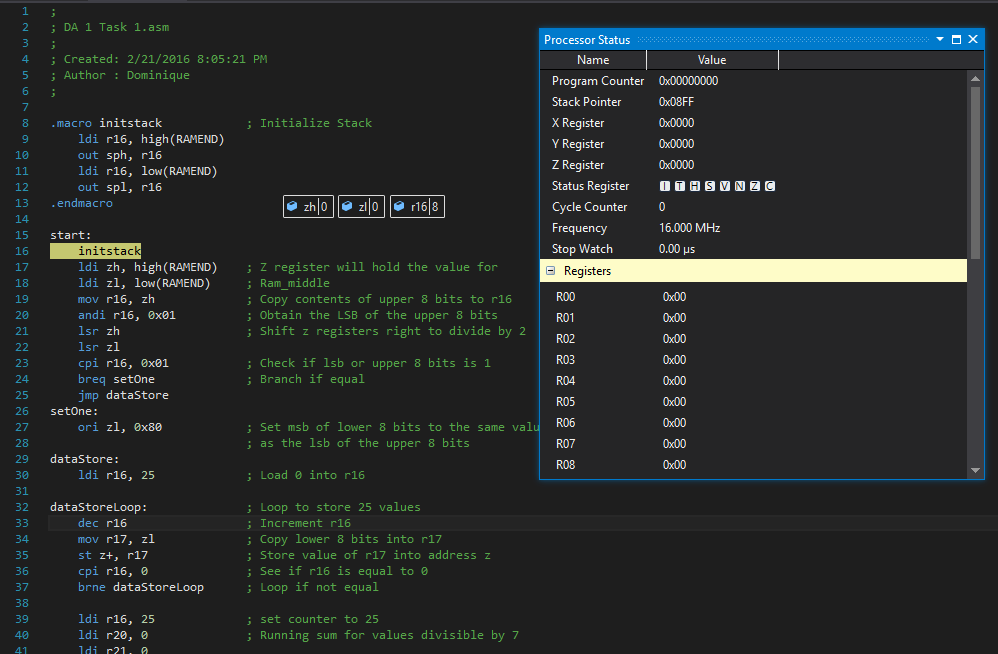
mov r7, r16 ; Copy the value into r7 in order to set the third bit to 1

done:

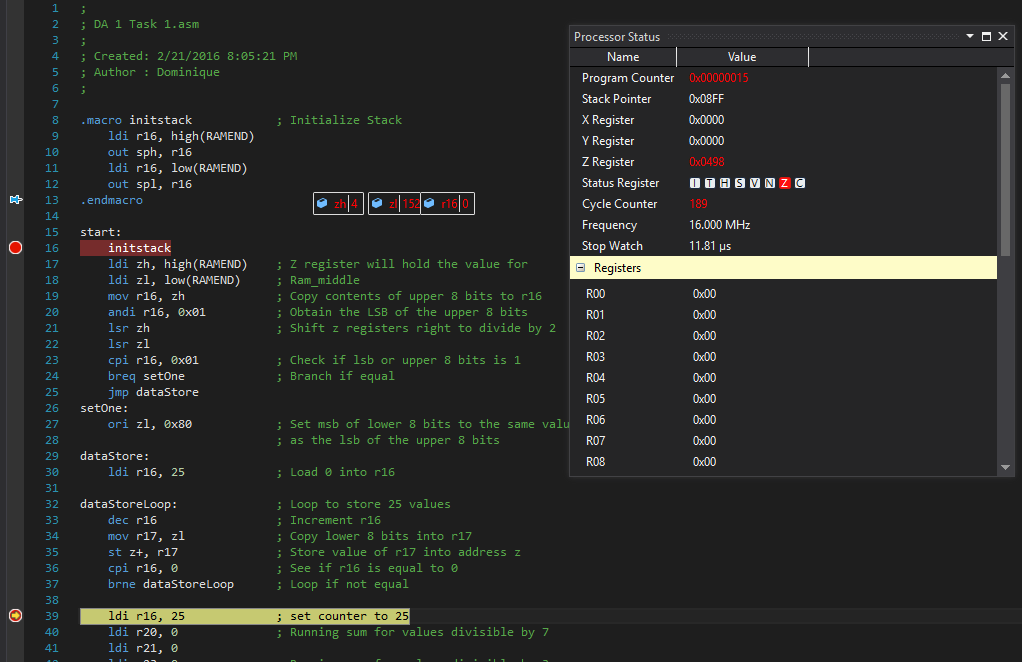
jmp done ; Continually loop at the end of the program

|  |  |  |  |
| --- | --- | --- | --- |
| 2. | SCREENSHOTS OF EACH TASK OUTPUT |  |  |

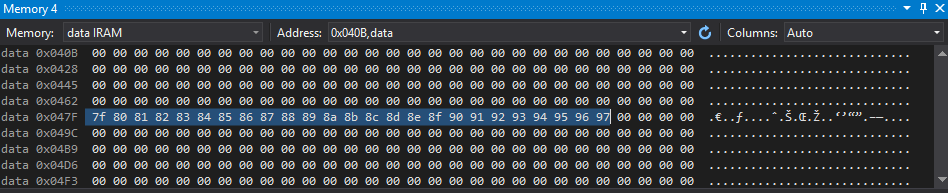
Program at the beginning of execution



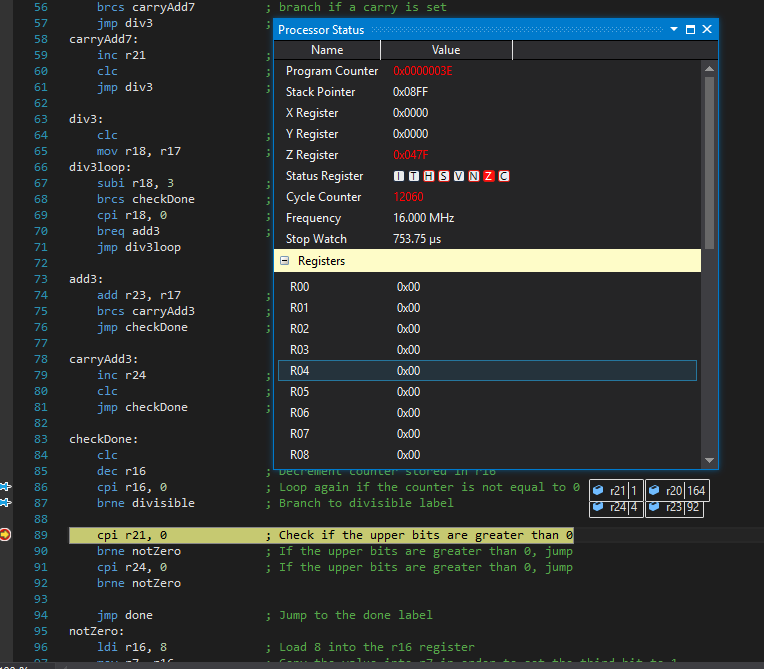
Program at the end of part A execution



Values Stored in RAM at the end of Part A execution

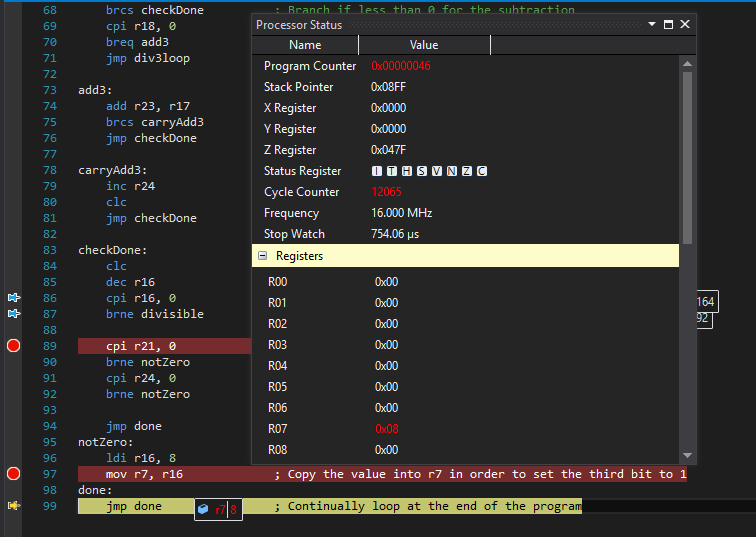


The Values of the registers at the end of determining whether a number was divisible by 3 or 7



Setting the 3rd bit of the r7 register to 1 at the end of the program

Execution time at 16 Mhz is also seen here



|  |  |  |  |
| --- | --- | --- | --- |
| 3. | FLOW CHART OF THE ASSEMBLY CODE |  |  |

START

Initialize the Stack

Load the Values into the Stack

Load Value from Stack for divisibility checking

Subtract 7 from a copy of the value

No

Is the value less than or equal to 0?

Less than 0.

Number is not divisible by 7.

Equal to 0

Number is divisible by 7

Subtract 3 from a copy of the value

No

Is the value less than or equal to 0?

Number is not divisible by 3.

Less than 0.

Equal to 0

Number is divisible by 3

|  |  |  |  |
| --- | --- | --- | --- |
| 4. | GITHUB LINK OF THE DA |  |  |
| <https://github.com/Anguian3/anguian3-submissions> | | | |

**Student Academic Misconduct Policy**

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

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

Dominique Anguiano