Andrew Wells

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)** |
| 0. | COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS |  |  |
| 1. | INITIAL CODE OF TASK 1/A |  |  |
| 2. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 2/B |  |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 3/C |  |  |
| 4. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 4/D |  |  |
| 5. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 5/E |  |  |
| 6. | SCHEMATICS |  |  |
| 7. | SCREENSHOTS OF EACH TASK OUTPUT |  |  |
| 8. | SCREENSHOT OF EACH DEMO |  |  |
| 9. | VIDEO LINKS OF EACH DEMO |  |  |
| 10. | GOOGLECODE LINK OF THE DA |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 0. | COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS |  |  |

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

; Andrew Wells CPE 301 DA1

; DA1T1.asm

;

; Created: 2/24/2016 9:26:24 PM

; Author: sirfe

;

; Macro to create the stack

.macro SET\_STACK ;Beginning of macro

ldi r16, LOW(RAMEND); Copy the lower 8 bits of the end of the RAM into R16

out spl, r16; Copy the address in r16 to the lower portion of the stack pointer

ldi r16, HIGH(RAMEND); Copy the higher 8 bits of the end of the RAM into R16

out sph, r16; Copy the address in r16 to the higher portion of the stack pointer

.endmacro

.cseg

.org 0

rjmp start; Skips over the interrupt portion of the memory

.org 0x20 ; starting point for the program

start:

SET\_STACK ; runs macro to create the stack

ldi r19, 25; Copy 25 into R19, keeps track of how many values have been obtained

ldi r20, 0 ; make sure R20 is 0 to start

ldi r21, 0 ; make sure R21 is 0 to start

ldi r23, 0 ; make sure R23 is 0 to start

ldi r24, 0 ; make sure R24 is 0 to start

ldi ZH, HIGH(RAMEND); Copy the High portion of the end of the RAM to ZH pointer/register

ldi ZL, LOW(RAMEND); Copy the Lower portion of the end of the RAM to the ZL pointer/register

; Divide by 2, to get to the middle of the RAM

lsr ZH; Shift the upper portion of Z to the right

ror ZL; Rotate the lower portion of Z to the right, brings in the carry(if any) from the upper portion

NEXT:

mov r17, ZL; Copy the value of the lower portion of the address in Z to R17

st Z+, r17; Store back the value into the RAM and increment Z to the next location

mov r16, r17; Copy the value in R17 to R16, to keep the value available for later calculations

DIV7:

subi r16, 7 ; Subtract the Value in R16 by 7

cpi r16, 7 ; Check to see if the value in r16 is 7

brsh DIV7 ; If the value is greater than or equal to 7, jump to DIV7 and continue to subtract

cp r16, r0; Compare the value in R16 to 0

brne NEXTDIV; If R16 is not 0 Skip to NEXTDIV and do not add value to total

add r20, r17; If value is divisible by 7 then add value to running total

adc r21, r21; Add R21 to itself, only increases if there is a carry from previous addition

NEXTDIV:

mov r16, r17; Copy the Value in R17 into R16

DIV3:

subi r16, 3 ; Subtract the Value in R16 by 3

cpi r16, 3 ; Check to see if the value in r16 is 3

brsh DIV3 ; If the value is greater than or equal to 3, jump to DIV3 and continue to subtract

cp r16, r0; Compare the value in R16 to 0

brne DIVDONE; If R16 is not 0 Skip to DIVDONE and do not add value to total

add r23, r17; If value is divisible by 3 then add value to running total

adc r24, r24; Add R24 to itself, only increases if there is a carry from previous addition

DIVDONE:

dec r19 ; Decrement R19 by 1

cpi r19, 0 ; Compare R19 with 0

brne NEXT ; If not zero return to beginning and get another value

cpi r21,0 ; Compare R21 with 0

brne SETREG ; If R21 is not 0 jump to SETREG

cpi r24,0 ; Compare R24 with 0

brne SETREG ; If R24 is not 0 jump to SETREG

HOLD:

rjmp HOLD ; Infinite Loop

SETREG:

ldi r16, 4 ; Copy 4 into R16

mov r7, r16; Copy 4 into R7, setting bit 3 to high

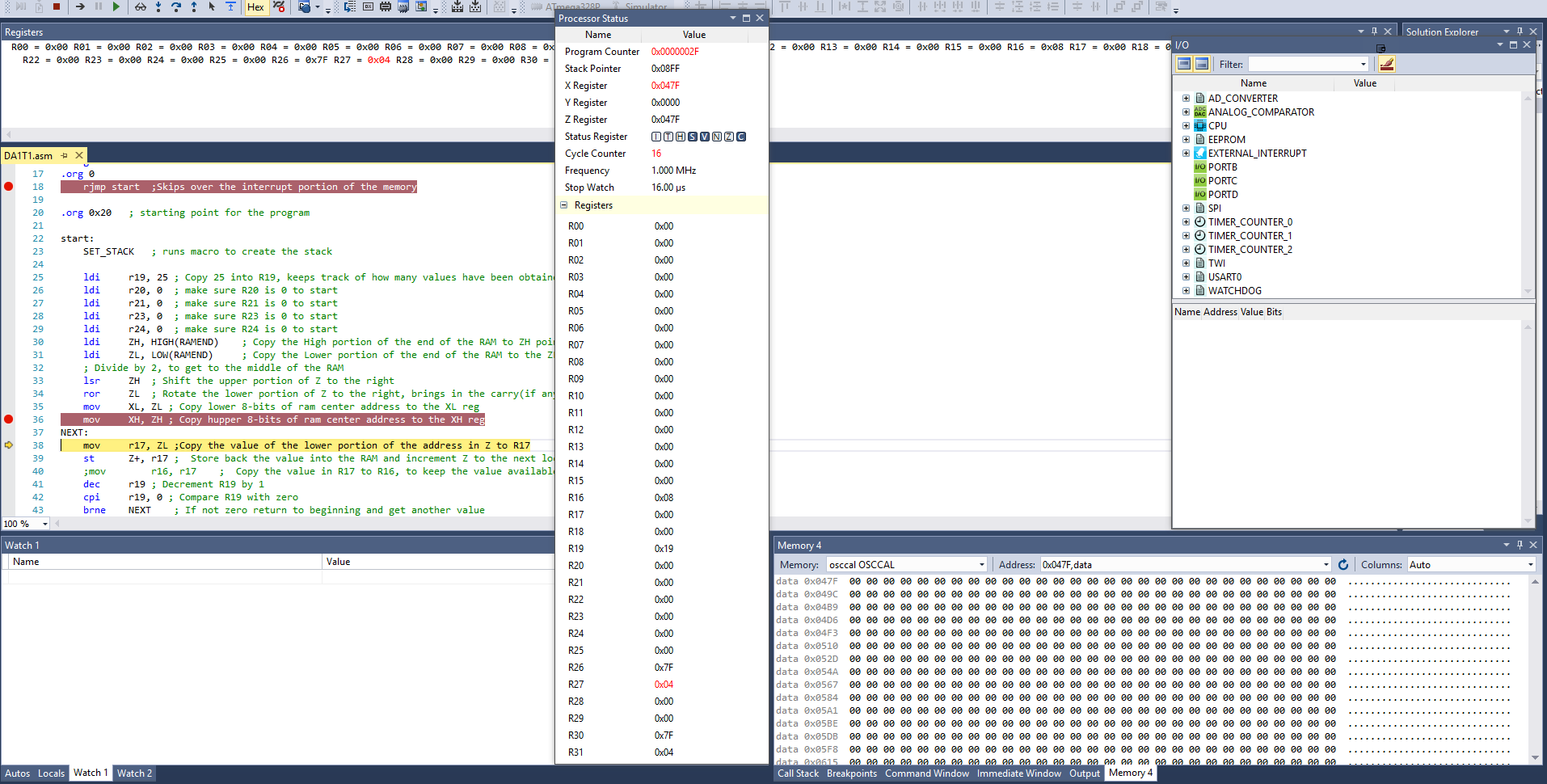
rjmp HOLD ; Jump to infinite loop

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

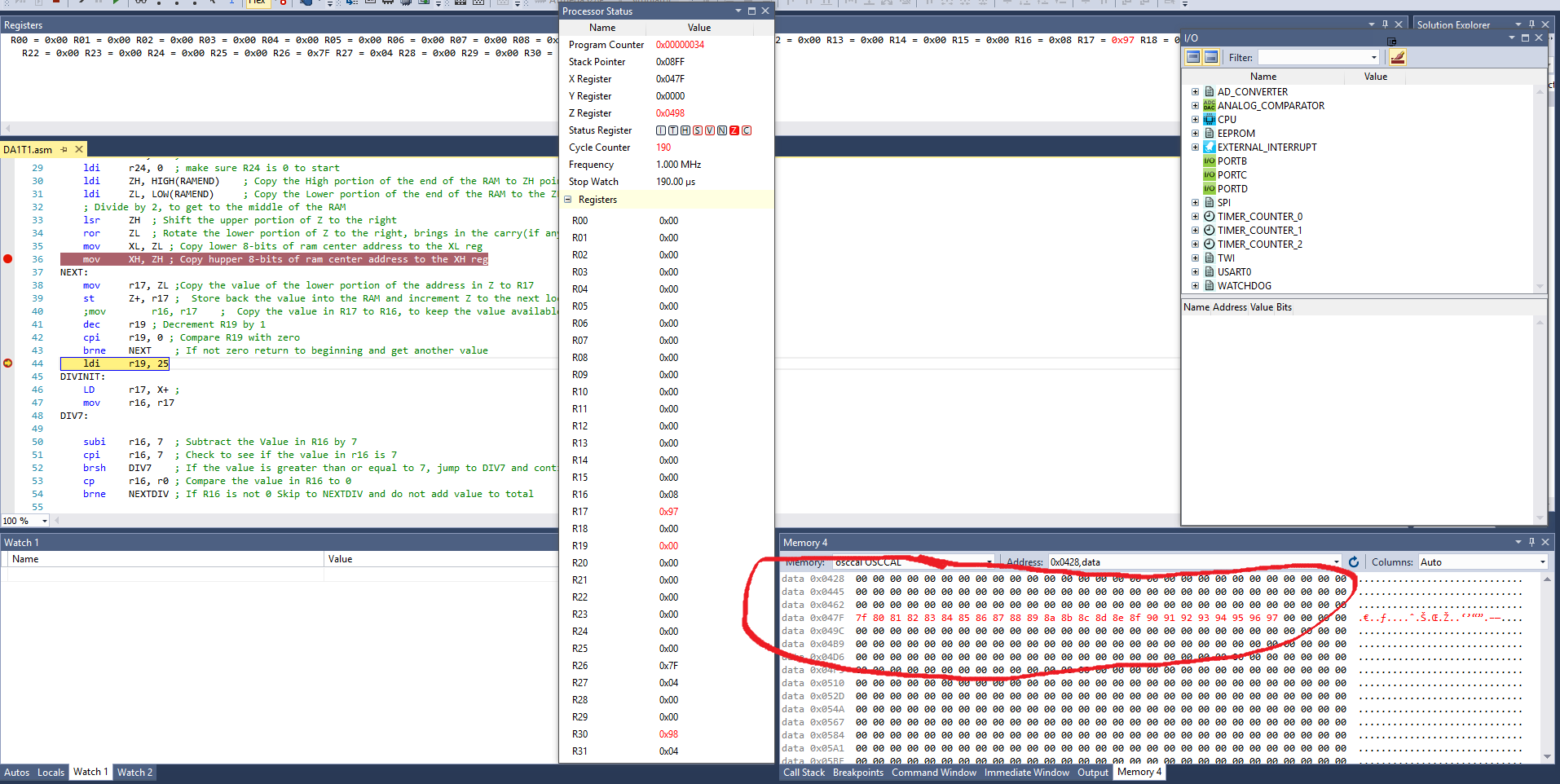
TASK 1/A:

Store 25 numbers starting from the RAMEND location. Capture the lower 8bits of the RAM MIDDLE = (RAMBEGIN + RAMEND)/2 address and use them as your values. You can increment or decrement rom RAM MIDDLE location to get the subsequent 24 numbers. Use the X/Y/Z registers to fill up the stack of 25 numbers.

**Before:**



**After:**



TASK1/B:

Parse through the 25 numbers and add all numbers divisible by 7 and place the

result in R20:21.

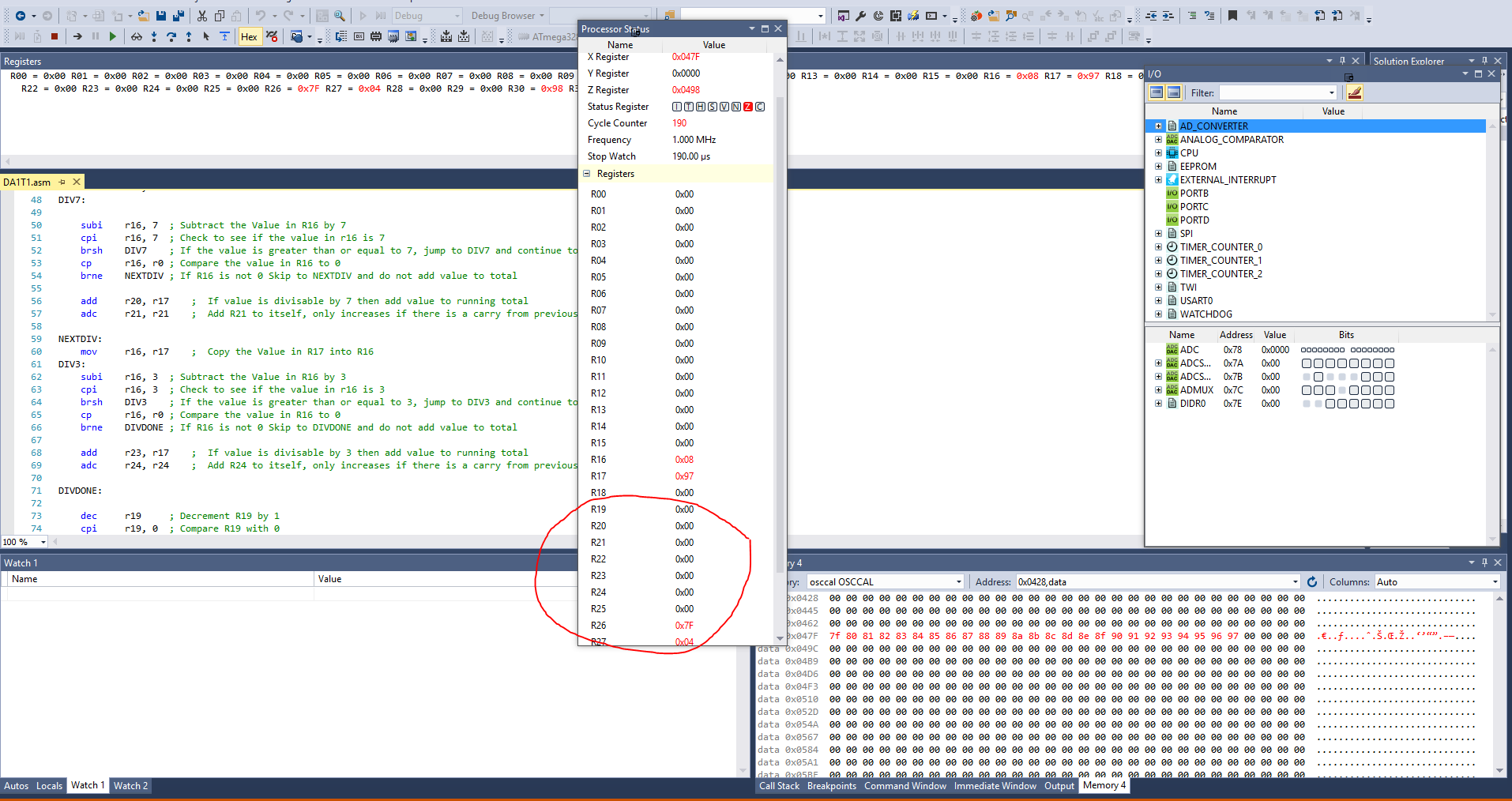
TASK1/C:

Parse through the 25 numbers and add all numbers divisible by 3 and place the

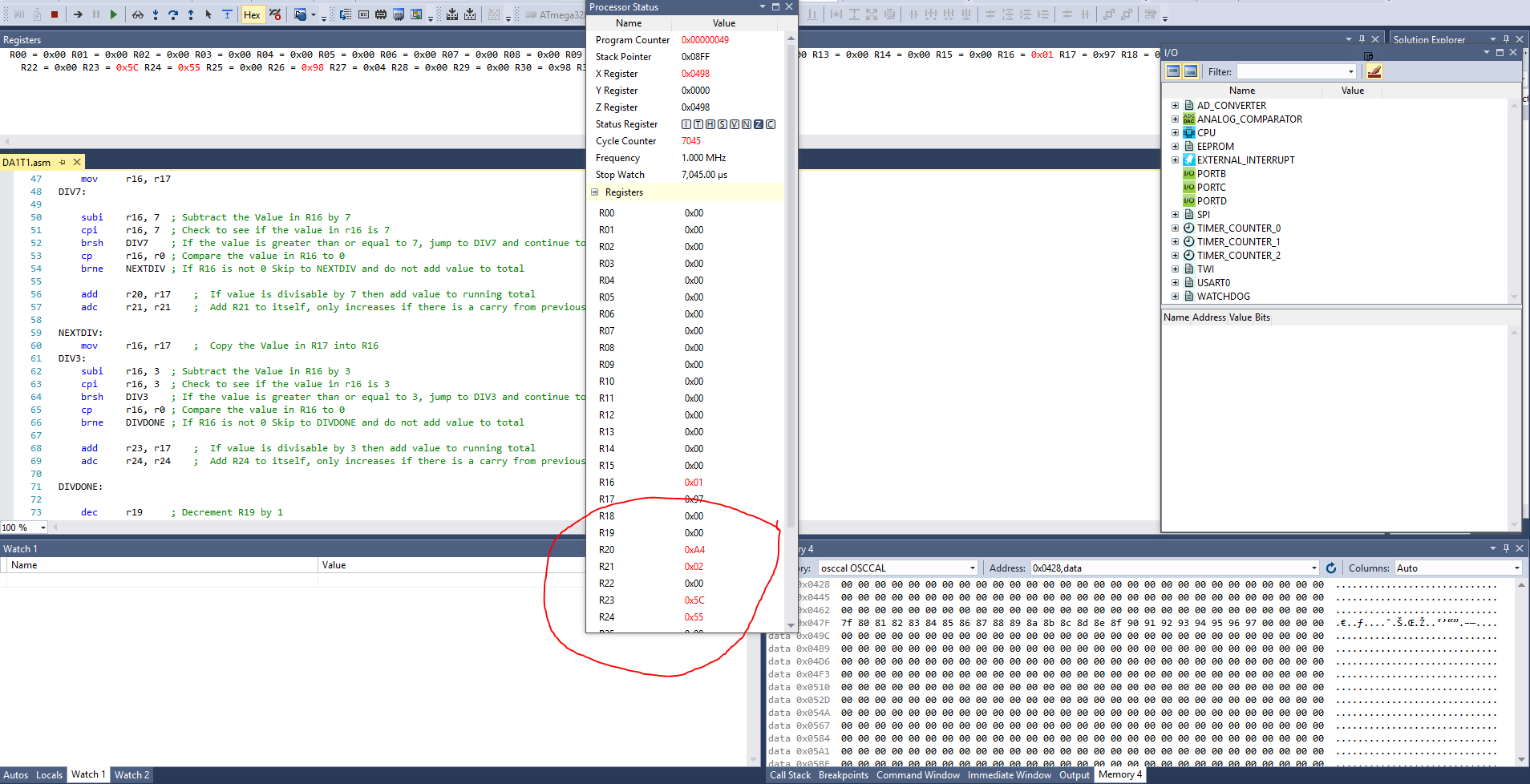
result in R23:24. Parsing of the numbers for task b and c has to be done

simultaneously.

**Before:**



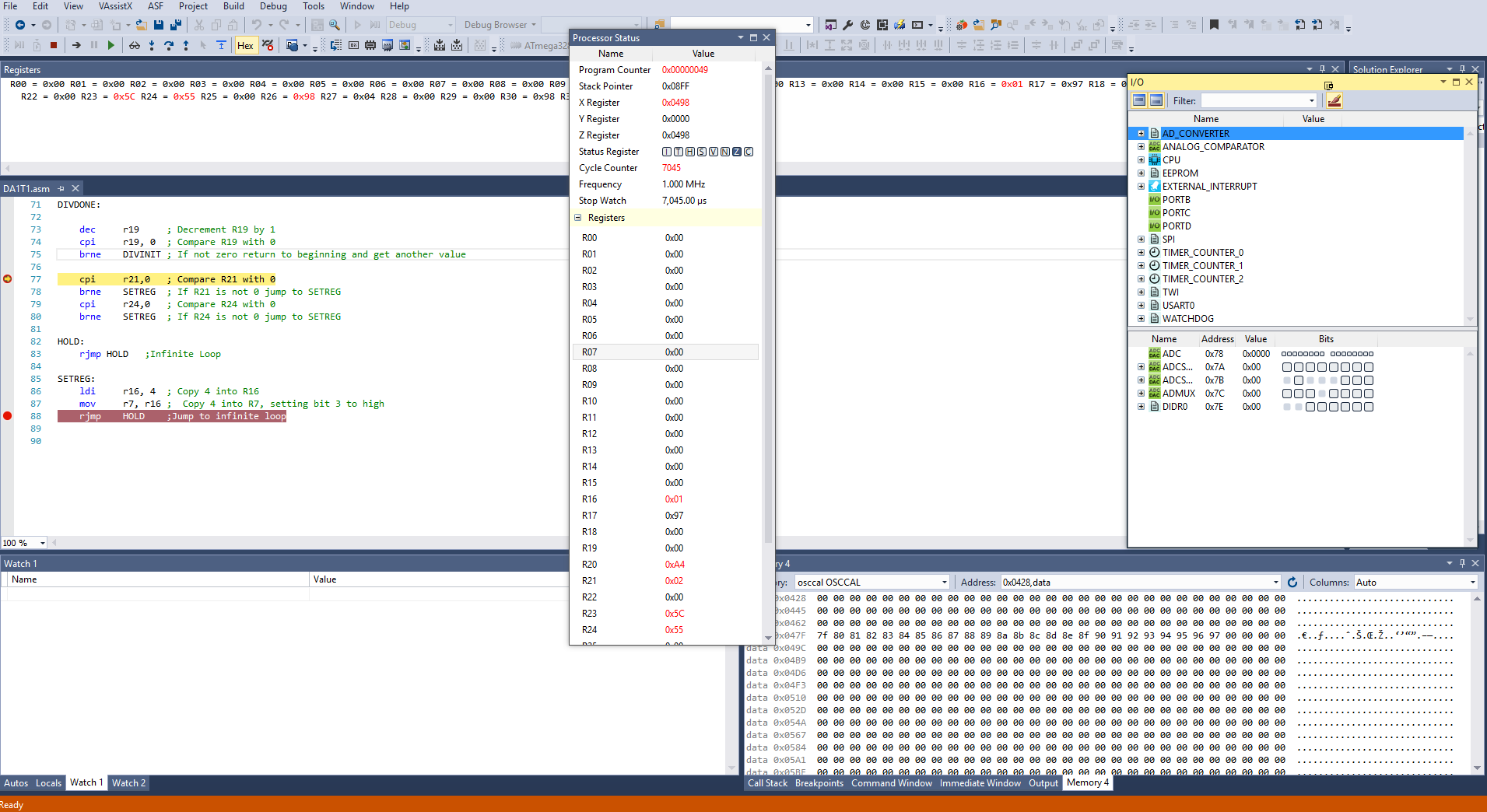
**After:**



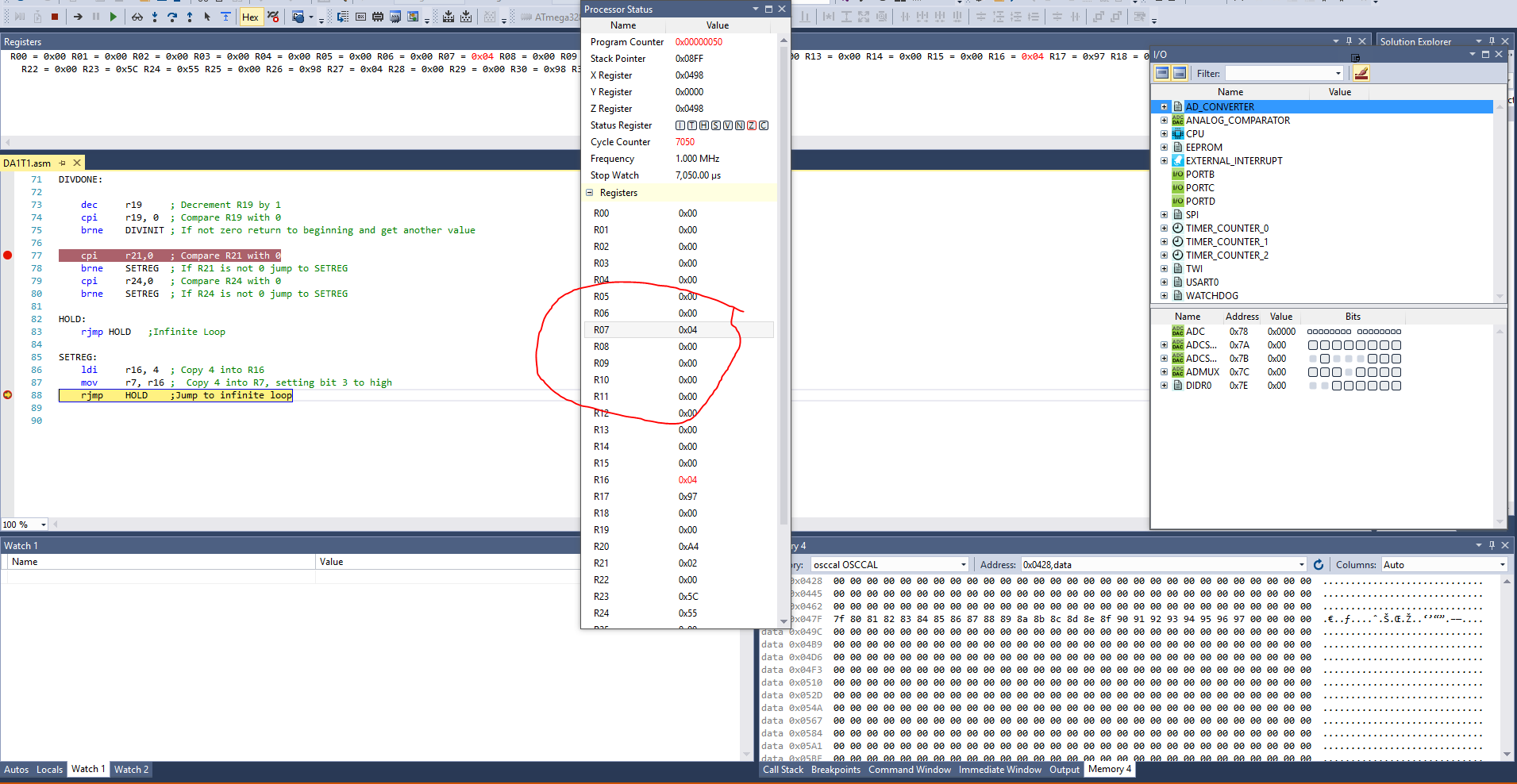
TASK1/D:

Check and set register R07.3 if the sum is greater than 16-bits.

**Before:**



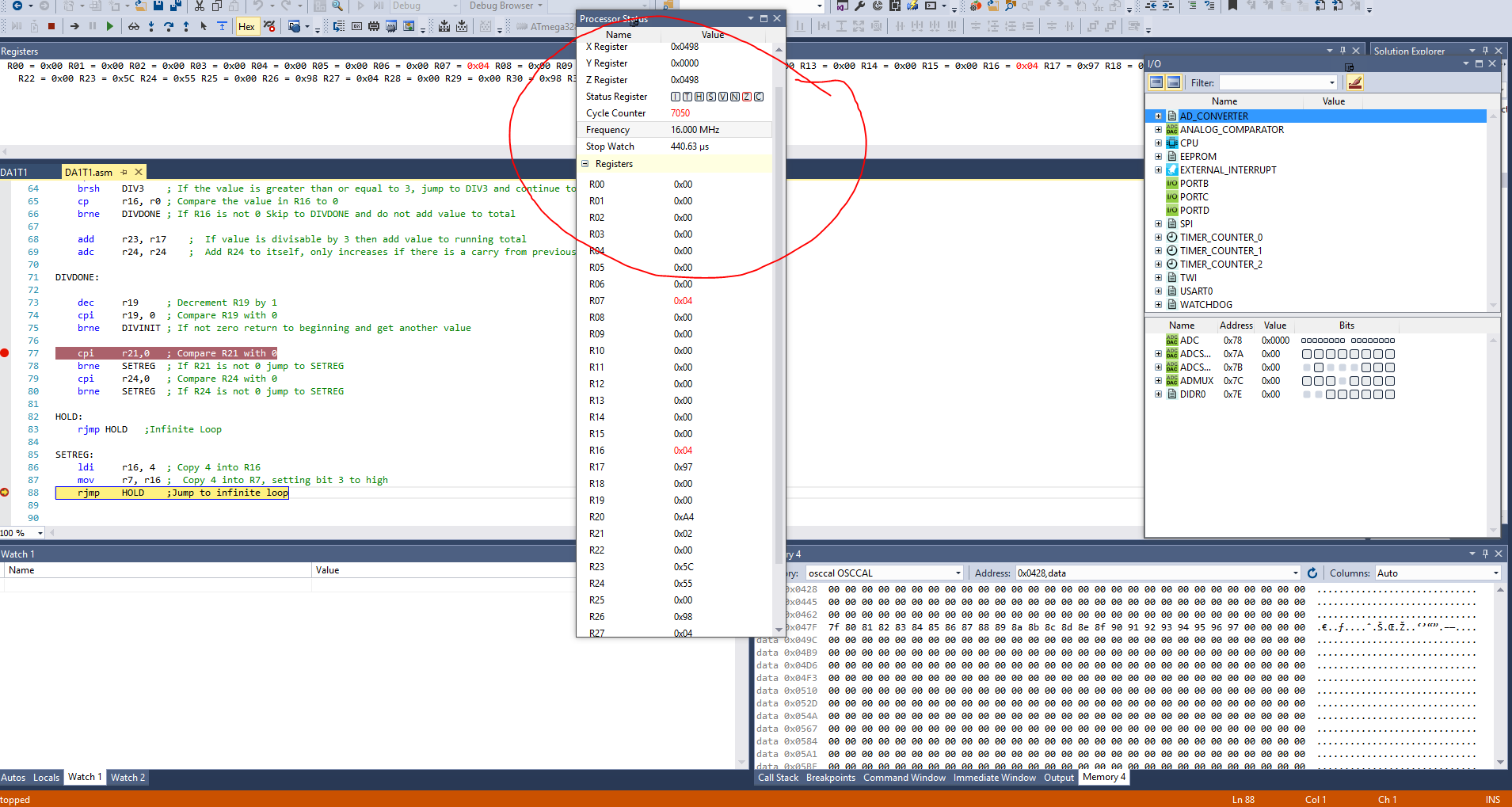
**After:**

****

TASK1/E:

Determine the execution time @ 16MHz/#cycles of your algorithm using the

simulation.



|  |  |  |  |
| --- | --- | --- | --- |
| 3. | Flowchart |  |  |

True

R16 >= 7

R16 = R16 - 7

Move value from R17 into R16

Load value from X into R17 Increment X

True

False

Is R19 = 0

Decrement R19

Store Value into Z memory location, increment Z

Copy lower portion of Z address into R17

Load RAMEND into Z Copy Z into X to save

Load 25 into R19

Create Stack

START

False

R16 = 0

False

True

R20 = R20 + R17

R21 = R21 + Carry

Move value from R17 into R16

R16 = R16 - 3

True

R16 >= 3

False

False

R16 = 0

True

R23 = R23 + R17

R24 = R24 + Carry

Decrement R19

False

R19 = 0

True

R21 != 0

False

R24 != 0

False

HOLD

Load 4 into R16

Move R16 into R7

|  |  |  |  |
| --- | --- | --- | --- |
| 9. | VIDEO LINKS OF EACH DEMO |  |  |
| No Video Taken | | | |
| 10. | GOOGLECODE LINK OF THE DA |  |  |
| https://github.com/Wellsa15/wellsa\_unlv | | | |

**Student Academic Misconduct Policy**

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

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

Andrew Wells