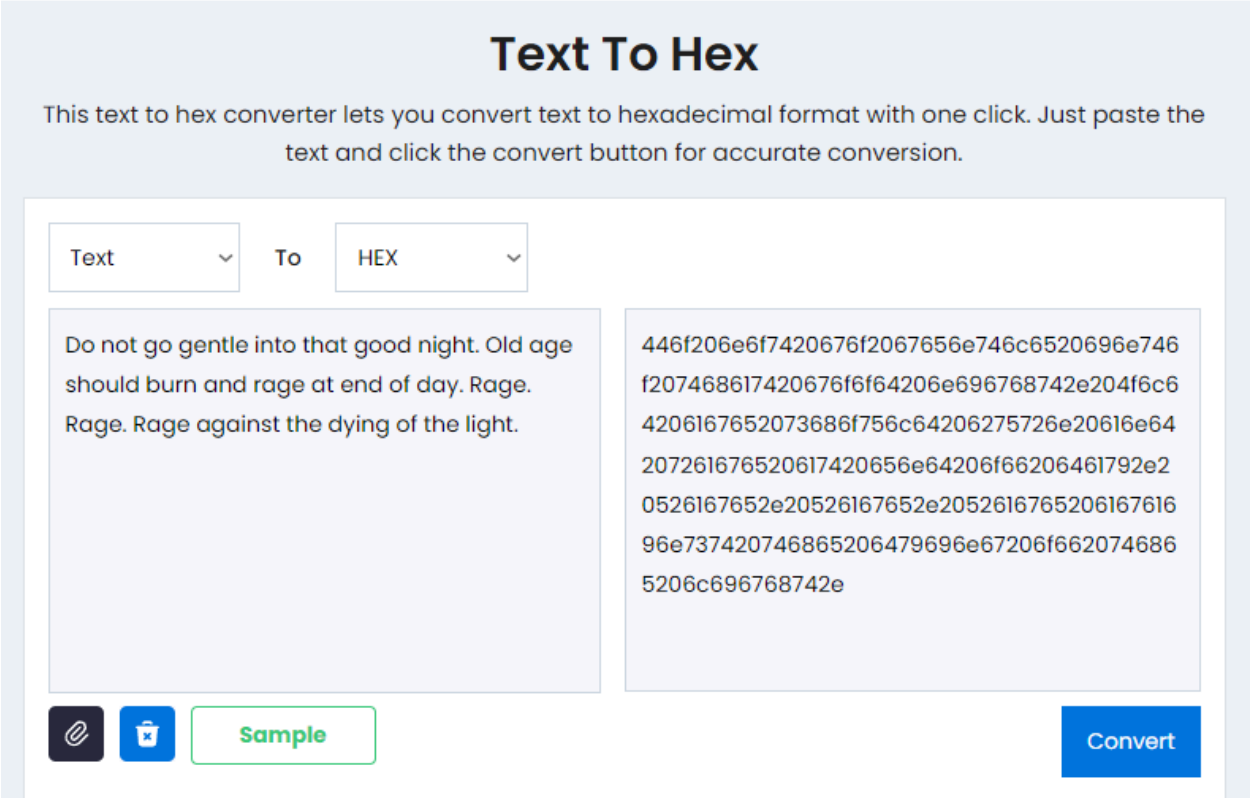
CPE 301
Spring 24
Design Assignment 1
Johnathan Widney

AVR Assembly code to take a sequence of sixteen 16-bit numbers, add the first ten as a 32-bit value, and store the value in both the middle of SRAM and EEPROM.

Below is a screenshot of my chosen text, that was generated into its respective hexadecimal value.

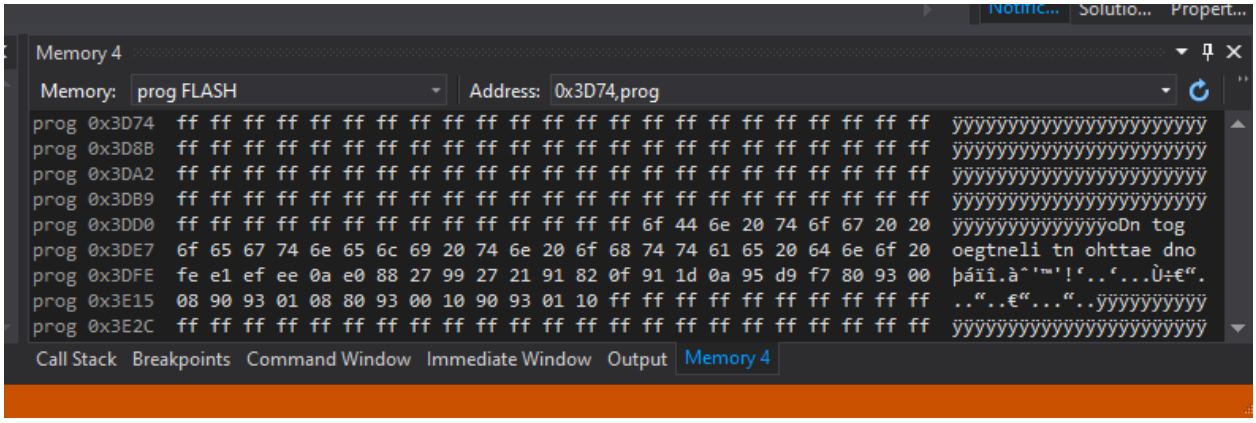


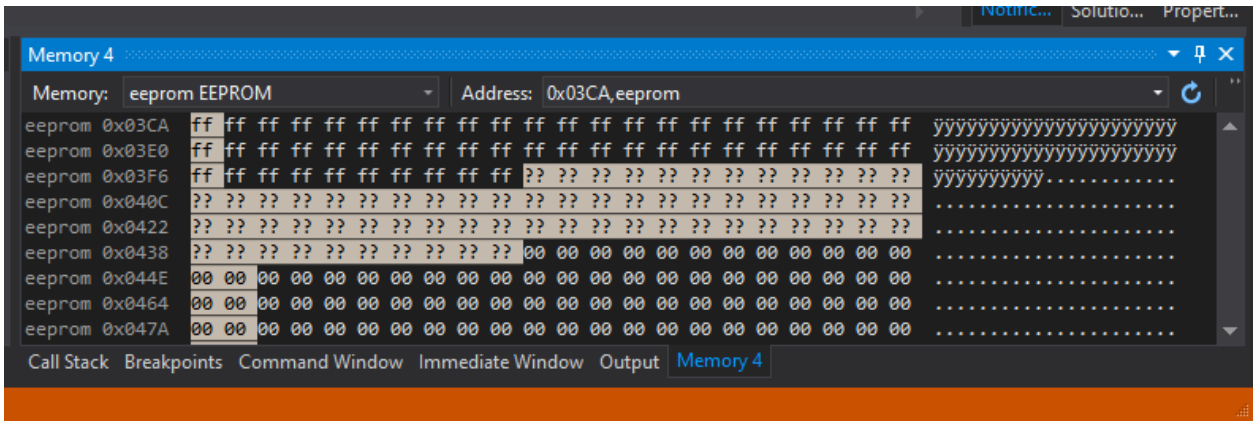
On the next page is the AVR assembly code with comments that details the operation.

```
;
; CPE301_DA-1.asm
;
; Created: 2/22/2024 10:20:14 PM
; Author : jdwid
;
; Design assignment 1, cpe 301 Johnathan Widney

.include "m328pdef.inc" ; Include ATmega328P definitions

.cseg ; Start of code segment in program memory
.org 0x1EEF ; Start address in program memory

; Defining the sequence of 16-bit numbers
Sequence:
    .dw 0x446f
    .dw 0x206e
    .dw 0x6f74
    .dw 0x2067
    .dw 0x6f20
    .dw 0x6765
    .dw 0x6e74
    .dw 0x6c65
    .dw 0x2069
    .dw 0x6e74
    .dw 0x6f20
    .dw 0x7468
    .dw 0x6174
    .dw 0x2065
    .dw 0x6e64
    .dw 0x206f
    ; ^ first sixteen 16-bits in my generated sequence

; Initializing Z pointer to point to the start of the sequence
    ldi ZH, high(Sequence) ; Store upper part of Z pointer register
    ldi ZL, low(Sequence) ; Store lower part of Z pointer register

; Calculating the running sum of the first ten numbers
    ldi R16, 10 ; Counter for ten iterations
    clr R24 ; Clear 32-bit sum (upper part)
    clr R25 ; Clear 32-bit sum (lower part)

 SumLoop:
    ld R18, Z+ ; Load next 16-bit number
```

```
    add R24, R18 ; Add to 32-bit sum (upper part)
    adc R25, R1 ; Add carry to 32-bit sum (lower part)
    dec R16 ; Decrement counter
    brne SumLoop ; Repeat until ten iterations are done

; Storing the 32-bit sum in the middle of SRAM (address 0x0800) - X pointer
    sts 0x0800, R24 ; Store upper part of sum in SRAM
    sts 0x0801, R25 ; Store lower part of sum in SRAM

; Storing the 32-bit sum in the middle of EEPROM (address 0x1000) - Y pointer
    sts 0x1000, R24 ; Stores upper part of sum in EEPROM
    sts 0x1001, R25 ; Stores lower part of sum in EEPROM



; Additional code to consider:

; Initializing X pointer (R26 - XH) and (R27 - XL)
; ldi R26, high(Sequence) ; Load upper part of X pointer
; ldi R27, low(Sequence)  ; Load lower part of X pointer

; Initializing Y pointer (R28 - YH) and (R29 - YL)
; ldi R28, high(Sequence) ; Load upper part of Y pointer
; ldi R29, low(Sequence)  ; Load lower part of Y pointer

; Loading data from pointers example:
; ld R18, X+ ; Load next 16-bit number from sequence (increment X)

; These are only 16 bit registers and therefore inefficient compared to
; using R24 and R25 which are 32-bit registers, and thus not needed in this code
```

Following are screenshots of simulation in the MicroStudio 7 environment for the code





The address locations were somehow altered, neither myself nor my colleagues are sure as to why. My suspicion is the IDE interpreter has defaults and other parameters that were not accounted for in the given instructions. As an example, the instructions asked to start memory at address location 0xBEEF, but this results in values and instructions being out of range. In order to mitigate this I had to edit the starting .org address to 0x1EEF for code to work.