

```

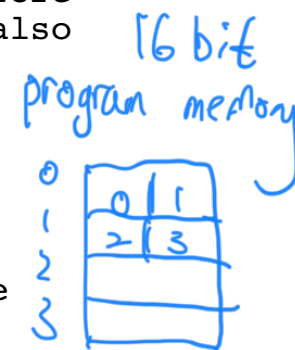
;
; CSC 230: Assignment 1
;
; YOUR NAME GOES HERE:
;   Date:
;
; This program generates each number in the Collatz sequence and stops at 1.
; It retrieves the number at which to start the sequence from data memory
; location labeled "input", then counts how many numbers there are in the
; sequence (by generating them) and stores the resulting count in data memory
; location labeled "output". For more details see the related PDF on connex.
;
; Input:
; (input) Positive integer with which to start the sequence (8-bit).
;
; Output:
; (output) Number of items in the sequence as 16-bit little-endian integer.
;
; The code provided below already contains the labels "input" and "output".
; In the AVR there is no way to automatically initialize data memory, therefore
; the code that initializes data memory with values from program memory is also
; provided below.

```

```

;
.cseg // specifies the start of code
.org 0 // set program counter (PC) to 0
    ldi ZH, high(init<<1) // load 8bit ; initialize Z to point to init
    ldi ZL, low(init<<1) // constant to r30
    lpm r0, Z+ // Z+ = *(ptr++) , LPM = load prog mem ; get the first byte
    sts input, r0 ; store it in data memory
    lpm r0, Z ; get the second byte
    sts input+1, r0 ; store it in data memory
    clr r0

```



8bit
Data memory
get 8bit
<<1 to get
8bit data.

```

;*** Do not change anything above this line ***

```

```

;****
; YOUR CODE GOES HERE:
;

```

```

;
; YOUR CODE FINISHES HERE
;****

```

```

;*** Do not change anything below this line ***

```

```

done:    jmp done

```

```

; This is the constant for initializing the "input" data memory location
; Note that program memory must be specified in double-bytes (words).

```

```

init:    .db 0x07, 0x00

```

```

; This is in the data memory segment (i.e. SRAM)
; The first real memory location in SRAM starts at location 0x200 on
; the ATmega 2560 processor. Locations below 0x200 are reserved for
; memory addressable registers and I/O

```

```

;
.dseg // start data segment
.org 0x200 // set SRAM address to 0x200
input:  .byte 2
output: .byte 2 // reserve 2 bytes in SRAM

```