1.

4.1

a.) Location counter assigns storage addresses to your program statements/instructions

c.) Object Code Produces bit patterns that are made to be loaded into the memory of the CUSP machine

d.) Assembler Directives are instructions that are used by the assembly, supply and control data.

f.) Symbol Table created by the compiler to keep track of the variables that were created/called.

2.

4.2

a.) .EQU @ can alter the location counter, .WORD also changes the LC. BLKW also changes the LC and .PAGE also alters the LC.

b.) .WORD, and .BLKW causes memory allocation

3.

Generates an error due to Y and Z not being loaded in yet. This means you cannot assign X to be Y * Z.

4.

4.5

a.)

Symbol	Value
SYM1	???
LAB2	???
SYM3	???

b.)

Symbol	Value
SYM1	???
LAB2	???
SYM3	???
LAB 4	???
JGE	LOOP

c.)

Symbol	Value
SYM1	$55
LAB2	$5B43DE
SYM3	$12C
LAB 4	???
JGE	LOOP

5.

Line	Address	Contents	Store
1			.EQU @203
2	$203-207	$000005	LDA VAR 1-2
3	$204	$4B0204	JGE LAB2
4			.EQU DEF, $102
5	$205	$302000	AND# DEF
6	$206	$400204	JMP LAB2
7			.EQU @$200

6.

4.7

c.)

VAR1	$203-207
LAB2	$204
DEF	$102

3.8

HW3,3.8.csp - ASIDE (CUSP)

File Edit CPU Help

```
1         LDA X ;loads accumulator with hexadecimal value
2         SHRA ;shifts bits to the right to check if the number is even or not
3         JNE $005 ;checks overflow at the least significant bit, if OV=0, then the bit is a 1. Goes to halt if its a 0
4         SHLA ; if bit is a 1, then we shift to the left and return to original place
5         JMP $006 ; jumps to Halt location
6         XOR X ;inverts the sign bit
7         HLT ; Stops the program
8         X: .WORD $0E0F03 ; Initial value
```

3.9

HW3.9.csp - ASIDE (CUSP)

File Edit CPU Help

```
1         JSR $E01 ;Takes in the critical value from user input
2         JSR $E01 ; Takes in the amount of data values we get from user input
3         STA NumWord
4         JLT $005 ; Checks to make sure amount of data values is number greater than 0, if it is we jump to line 8
5         JMP $008
6         LDA# $000 ; If it is not greater than 0, we load 0 into the accumulator
7         JSR $E00 ; prints out 0 for the a wrong input
8         JMP $001 ; jumps back to $001 to ask the data values again
9         STA NumWord ;Stores whatever is in the accumulator to NumWord
10        LDA M ;Loads in M
11        JSR $E01 ;Asks for user input
12        STA Data ;Stores user input to one of the data slots
13        LDA M ; Loads M into the accumulator again
14        CMA# target ; compares if M is still less than target
15        JMP $018 ; if M is greater than target then jumps out of the loop
16        ADA# step ; if it is not M greater than target, it adds 1 to M
17         STA M ;stores new value for M
18        JMP $009 ;jumps back to the beginning
19        LDA Data ;load in the values from data
20        LDA Critical ; Load in the critical value
21        CMA# Data ;compare values
22        INC $205 ;increments if its less than, so increments counter1
23        JMP $024 ;jumps to the end of the loop
24        INC $206 ;increments the counter2 if it is greater than critical value
25        JMP $018 ; jump to beginning of the loop
26        LDA Counter1 ;load counter1
27        JSR $E00 ;print output
28        LDA Counter2 ;load counter 2
29        JSR $E00 ;print out counter2
30         HLT
31        .EQU @,200
32    Critical: .WORD $E01
33    NumWord: .WORD $001
34    Data: .blkw NumWord
35    .EQU target, NumWord
36    .EQU step, 1
37    M: .WORD 0 ; iterator for the first loop which stores the values in
38    N: .WORD 0 ; operator for the second loop which compares the values
39    Counter1: .WORD 0 ; Less than counter
40    Counter2: .WORD 0 ; greater than counter
```

4.13

```
1          .EQU PUT_STR, $E05
2          .EQU PUT_NL, $E06
3          LDA M ;loads loop variable
4          STA M ;stores M
5          LDS# $E00
6          PSH# STRING_LENGTH ;print the string
7          PSH# STRING
8          JSR PUT_STR
9          ADS# 2
10         JSR $E01 ;gets User input
11         STA Number ;stores user input
12         CMA# Target2 ; compares number against target 2 which is 0
13         JMP $023 ;jumps to halt
14         JSR PUT_NL
15         LDA N ; new loop variable
16         LDA Number ; load in the number
17         SHLA ;shift the bits over
18         JNO $017 ; throws a flag
19         JSR $E00 ; prints out the OV flag
20         LDA N
21         STA N ; stores and loads new loop variable
22         CMA# Target ; compare N against target
23         JSR PUT_NL ; puts a new line
24         JMP $000; restarts the whole user prompts again
25         JMP $012 ;jumps back to line 10 to restart the loop
26         HLT
27 STRING: .CHAR 'Enter number(0 = quit)', STRING_LENGTH ;Defines the string
28 M: .WORD 0
29 N: .WORD 0
30 Step: .WORD 1
31 Target: .WORD 23
32 Target2: .WORD 0
33 .EQU Number, 0
```