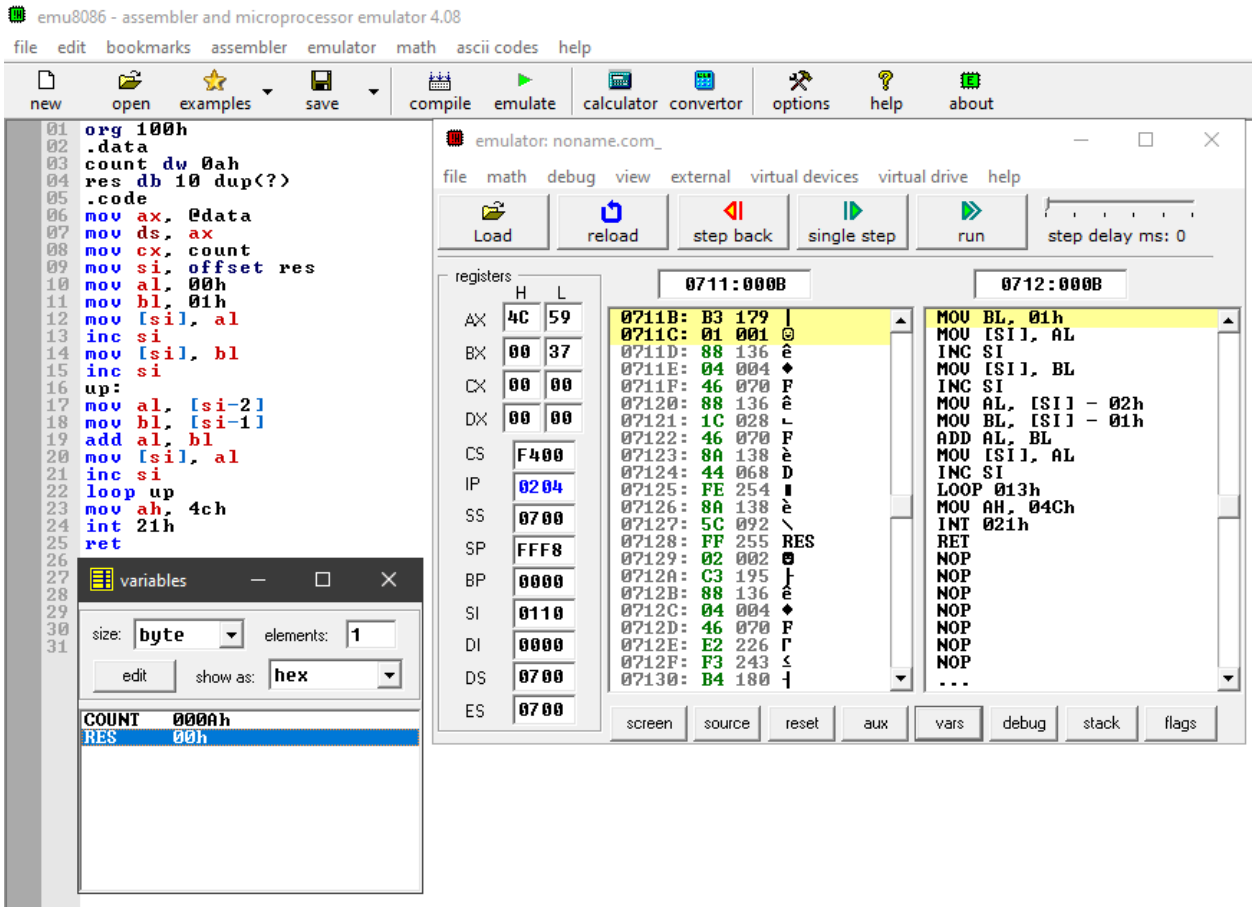
Experiment 3
Class: SE Comp
Year: 2020-21
Performed by: Danyl Fernandes, 72

Ascending order:

Code:

```
org 100h
.data
     count dw 0ah
     res db 10 dup(?)
.code
     mov ax, @data
     mov ds, ax
     mov cx, count
     mov si, offset res
     mov al, 00h
     mov bl, 01h
     mov [si], al
     inc si
     mov [si], bl
     inc si
up:
     mov al, [si-2]
     mov bl, [si-1]
     add al, bl
     mov [si], al
     inc si
     loop up
     mov ah, 4ch
     int 21h
ret
```

Output:

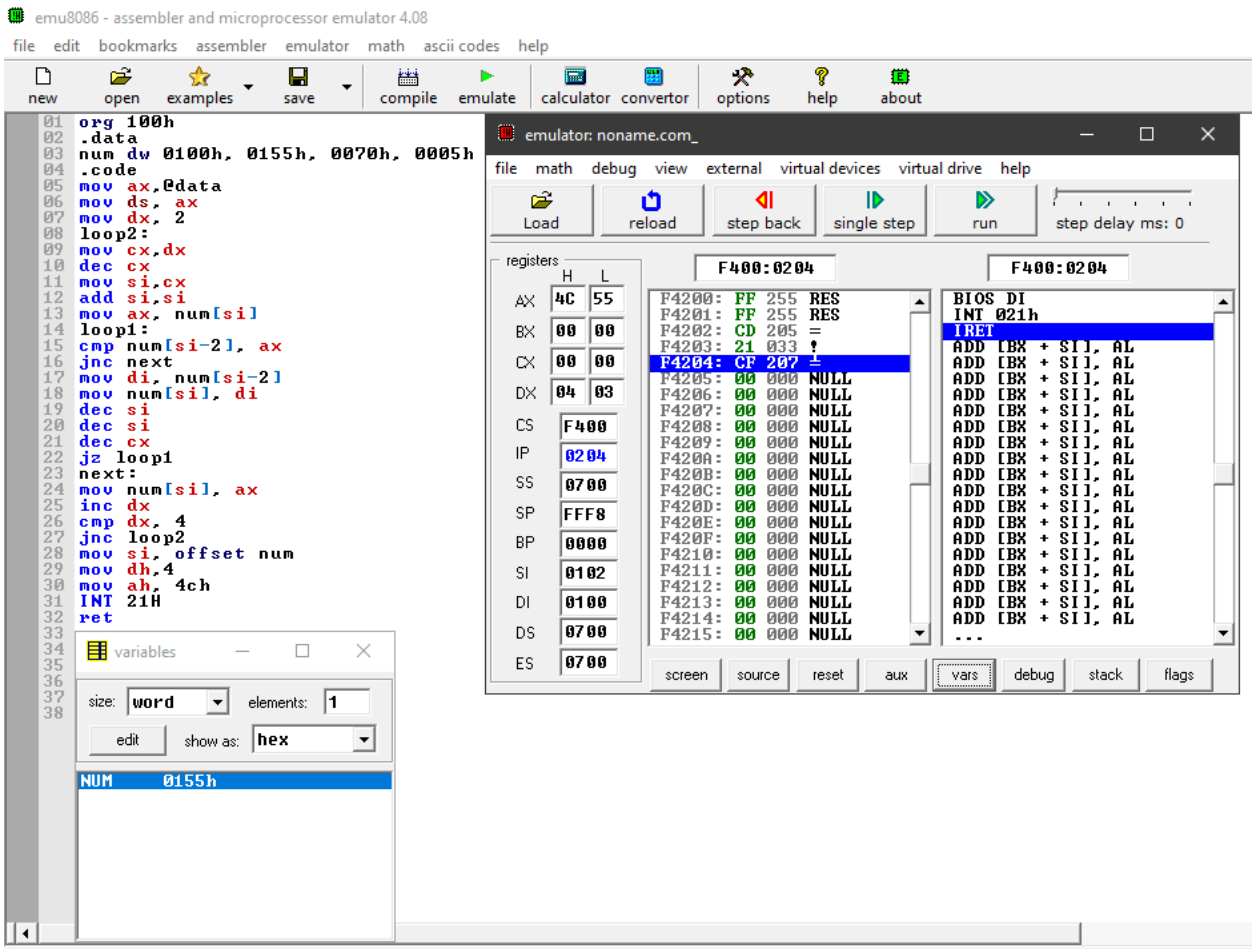

emu8086 - assembler and microprocessor emulator 4.08
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
01 org 100h
02 .data
03 count dw 0ah
04 res db 10 dup(?)
05 .code
06 mov ax, @data
07 mov ds, ax
08 mov cx, count
09 mov si, offset res
10 mov al, 00h
11 mov bl, 01h
12 mov [si], al
13 inc si
14 mov [si], bl
15 inc si
16 up:
17 mov al, [si-2]
18 mov bl, [si-1]
19 add al, bl
20 mov [si], al
21 inc si
22 loop up
23 mov ah, 4ch
24 int 21h
25 ret
26
27
28
29
30
31
variables
size: byte elements: 1
edit show as: hex
COUNT 000Ah
RES 00h
emulator: noname.com_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
registers
H L
AX 4C 59
BX 00 37
CX 00 00
DX 00 00
CS F400
IP 0204
SS 0700
SP FFF8
BP 0000
SI 0110
DI 0000
DS 0700
ES 0700
0711:000B
0712:000B
0711B: B3 179
0711C: 01 001
0711D: 88 136
0711E: 04 004
0711F: 46 070 F
07120: 88 136
07121: 1C 028
07122: 46 070 F
07123: 8A 138
07124: 44 068 D
07125: FE 254
07126: 8A 138
07127: 5C 092 \
07128: FF 255 RES
07129: 02 002
0712A: C3 195
0712B: 88 136
0712C: 04 004
0712D: 46 070 F
0712E: E2 226
0712F: F3 243
07130: B4 180
MOV BL, 01h
MOV [SI], AL
INC SI
MOV [SI], BL
INC SI
MOV AL, [SI] - 02h
MOV BL, [SI] - 01h
ADD AL, BL
MOV [SI], AL
INC SI
LOOP 013h
MOV AH, 04Ch
INT 021h
RET
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
screen source reset aux vars debug stack flags

Descending order:

Code:

```
org 100h
.data
     num dw 0100h, 0155h, 0070h, 0005h
.code
     mov ax,@data
     mov ds, ax
     mov dx, 2
loop2:
     mov cx,dx
     dec cx
     mov si,cx
     add si,si
     mov ax, num[si]
loop1:
     cmp num[si-2], ax
     jnc next
     mov di, num[si-2]
     mov num[si], di
     dec si
     dec si
     dec cx
     jz loop1
next:
     mov num[si], ax
     inc dx
     cmp dx, 4
     jnc loop2
     mov si, offset num
     mov dh,4
     mov ah, 4ch
     INT 21H
ret
```

Output:



Conclusion:

We successfully wrote assembly language programs to arrange blocks of data in ascending and descending order

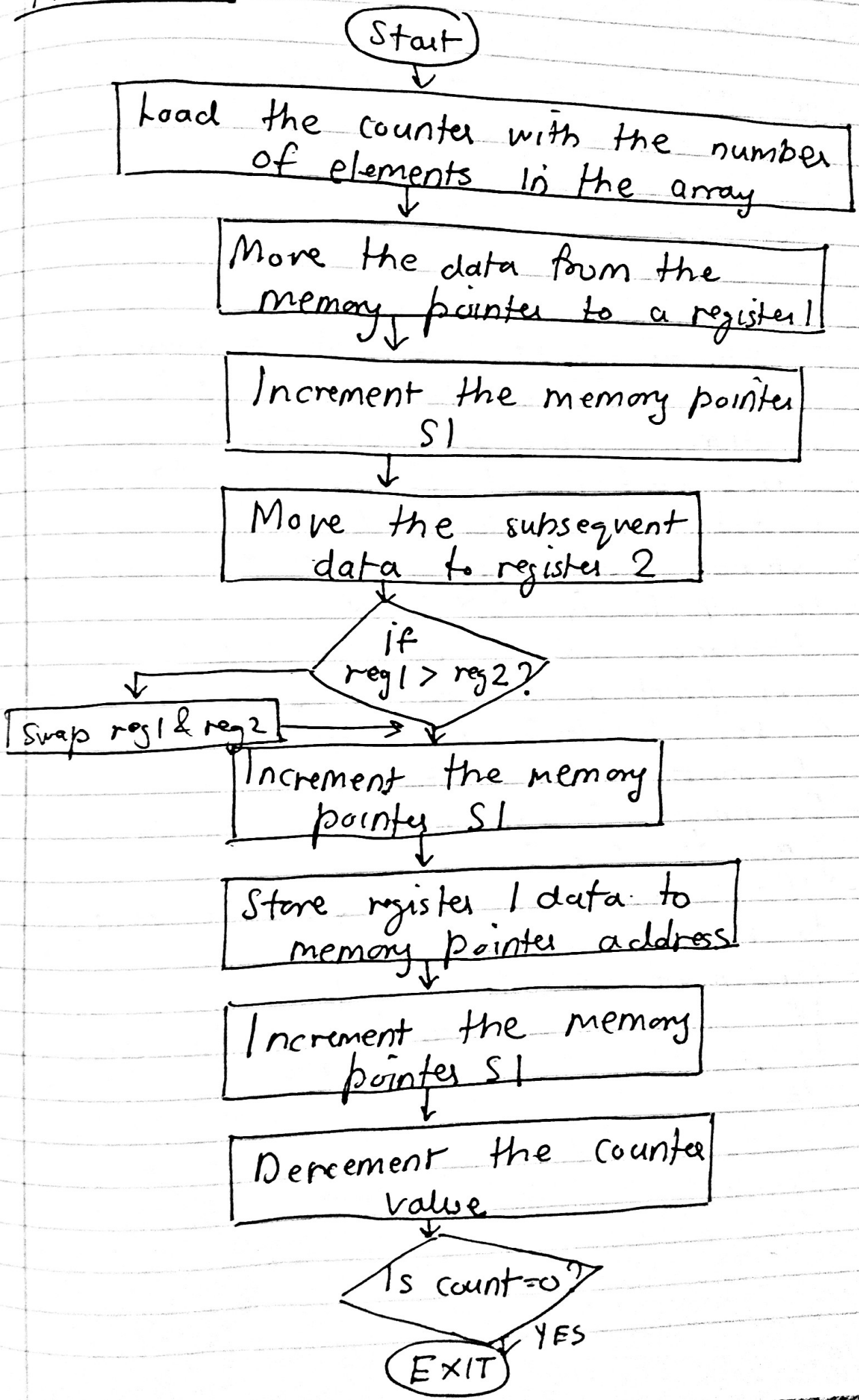
Exp 03

Aim: To arrange the block of data in Ascending order.

Algorithm:

- Initialize counter 1
- Initialize point 1 to first memory location
- Initialize point 2 to second memory location & Initialize counter 2
- Move contents at pointer 1 to AX & Move contents at pointer 2 to BX
- Compare values of AX & BX & Jump to step 12 if AX < BX
- Exchange values of AX & BX
- Set pointer 1 to data in AX and set pointer 2 point to data in BX
- Increment both pointers by 2
- Decrement 2 when not equal to zero jump to step 5
- If counter 2 is not equal to zero jump to step 2
- Halt.

Flow chart:


Start
Load the counter with the number of elements in the array
Move the data from the memory pointer to a register 1
Increment the memory pointer SI
Move the subsequent data to register 2
if reg1 > reg2?
Swap reg1 & reg2
Increment the memory pointer SI
Store register 1 data to memory pointer address
Increment the memory pointer SI
Decrement the counter value
Is count=0?
YES
EXIT

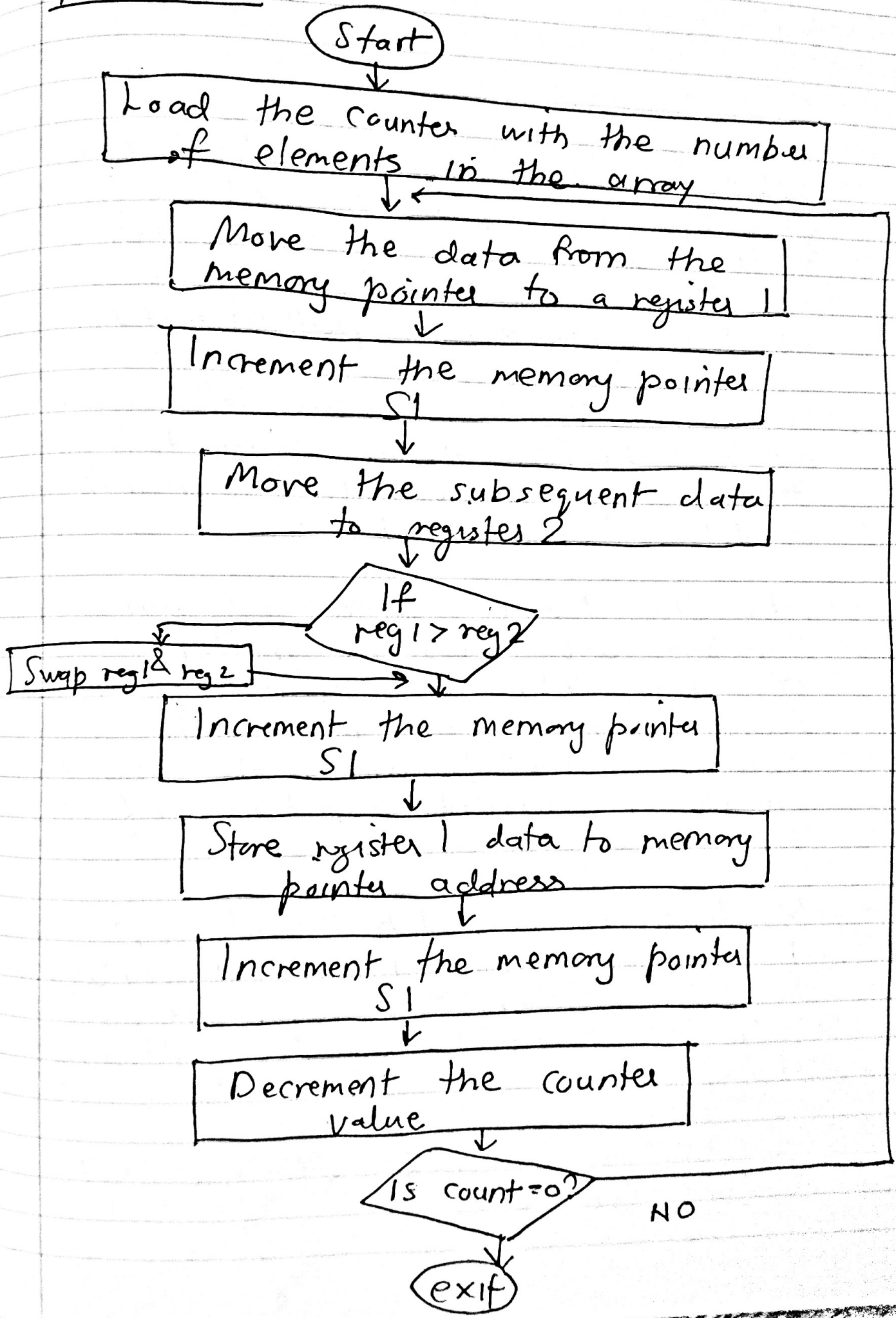
b) Descending order

Aim: To arrange the block of data in Descending order

Algorithm:

- Initialize counter 1
- Initialize point 1 to frist memory location
- Initialize point 2 to second memory location & Initialize counter 2
- Move contents at pointer 1 to AX & Move contents at pointer 2 to BX
- Compare values of AX & BX and Jump to step 12 if AX > BX
- Exchange values of AX & BX
- Set pointer 1 pointer to data in AX & set pointer 2 point to data in BX
- Increment both pointers 2
- Decrement 2 when not equal to zero jump to step 5
- If counter 2 is not equal to zero, jump to step 5
- Decrement counter 1 by 1
- If counter 1 is not equal to zero jump to step 2
- Halt

Flow Chart :


Start
Load the counter with the number of elements in the array
Move the data from the memory pointer to a register 1
Increment the memory pointer SI
Move the subsequent data to register 2
If reg1 > reg2
Swap reg1 & reg2
Increment the memory pointer SI
Store register 1 data to memory pointer address
Increment the memory pointer SI
Decrement the counter value
Is count=0?
NO
exit