

Name: Jigar Siddhpura

SAPID: 60004200155

DIV: C/C2

Branch: Computer Engineering

POA EXPERIMENT 6

60004200155
Jigar Siddhpura

Experiment 6: Sorting Numbers in 8086

Aim: Implement program to sort numbers in ascending / descending order.

Theory:

Bubble sort is implemented in 8086 microprocessor to arrange an array of values. Here, elements are compared to next adjacent element and swapped if they are in wrong order; until the array is sorted. The assembly code shows both ascending & descending sorting.

Ascending:

Here, array (segment) is defined in DATA segment & iterated through using loops. Outer loop 'L1' iterates through array & inner loop 'L2' is used for comparison. If current element is greater than next element, swapping is performed using XCHG instruction. This continues until entire array is sorted.

Descending: Here

Here, the code is almost same as above, however the primary difference is on Conditional Jump instructions. Here, swap is performed if current element is less

than the next .

Here instructions used are - MOV, LEA (load effective address), DEC, COMP (compare), JNZ, JC, XCHG (swap operation)

Here, code segment contains instructions whereas data segment is used to define array. In code, DATA refers to 0710h address, CH refers to counter, ^{ns} LEA SI, NI stores starting address of NI. Under the hood; CMP performs subtraction to detect the greater element. JC means Jump if Carry i.e. if carry flag is set, jump to particular procedure. JNZ means Jump if not zero i.e. jump if flag is non-zero.

Conclusion: Here, we perform sorting in both possible orders on 8086 microprocessor using a bubble sort ~~★~~ with the help of instruction set of 8086.

Code :

1. Sorting in Ascending Order :

DATA SEGMENT

N1 DB 10h,14h,7h,8h,98h,19h,34h,5h, 98h

SIZE equ \$ - N1

ENDS

CODE SEGMENT

START:

MOV AX,DATA

MOV DS,AX

MOV CH,SIZE

L1:

LEA SI,N1

MOV CL,

SIZE DEC CL

L2:

MOV AL, [SI]

MOV BL, [SI+1]

CMP AL,BL

JC DOWN

MOV DL,[SI+1]

XCHG [SI],DL

MOV [SI+1],DL

DOWN:

INC SI

DEC CL

JNZ L2

DEC CH

JNZ L1

CODE ENDS

END START

- Here the elements are sorted in ascending order in N1 data segment

Before Sorting -

The screenshot shows the POA6.1 emulator interface. The registers window on the left shows the current state of the registers. The main window displays the assembly code and the N1 data segment. The N1 data segment is shown in the variables window at the bottom, with the following values: 10h, 14h, 07h, 08h, 98h, 19h, 34h, 05h.

Registers:

Register	H	L
AX	00	00
BX	00	00
CX	00	38
DX	00	00
CS	0711	
IP	0000	
SS	0710	
SP	0000	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

Assembly Code:

```

07110: B8 184 7 MOV AX, 00710h
07111: 10 016 MOV DS, AX
07112: 07 007 BEEP MOV CH, 09h
07113: 8E 142 MOV SI, 00000h
07114: D8 216 MOV CL, 09h
07115: B5 181 DEC CL
07116: 09 009 TAB MOV AL, [SI]
07117: BE 190 MOV BL, [SI] + 01h
07118: 00 000 CMP AL, BL
07119: 00 000 JB 01Fh
0711A: B1 177 MOV DL, [SI] + 01h
0711B: 09 009 XCHG [SI], DL
0711C: FE 254 MOV [SI] + 01h, DL
0711D: C9 201 INC SI
0711E: 8A 138 DEC CL
0711F: 04 004 JNE 0Eh
07120: 8A 138 DEC CH
07121: 5C 092 JNE 07h
07122: 01 001 NOP
07123: 3A 058 NOP
07124: C3 195 ...

```

Variables:

size: byte elements: 8 show as: hex

N1 10h, 14h, 07h, 08h, 98h, 19h, 34h, 05h

After sorting -

The screenshot shows the POA6.1 emulator interface after sorting. The registers window on the left shows the current state of the registers. The main window displays the assembly code and the N1 data segment. The N1 data segment is shown in the variables window at the bottom, with the following values: 05h, 07h, 08h, 10h, 14h, 19h, 34h, 98h.

Registers:

Register	H	L
AX	07	98
BX	00	98
CX	00	00
DX	00	98
CS	0711	
IP	003C	
SS	0710	
SP	0000	
BP	0000	
SI	0000	
DI	0000	
DS	0710	
ES	0700	

Assembly Code:

```

07149: 90 144 E NOP
0714A: 90 144 E NOP
0714B: 90 144 E NOP
0714C: 90 144 E NOP
0714D: 00 000 NULL
0714E: 00 000 NULL
0714F: 00 000 NULL
07150: 00 000 NULL
07151: 00 000 NULL
07152: 00 000 NULL
07153: 00 000 NULL
07154: 00 000 NULL
07155: 00 000 NULL
07156: 00 000 NULL
07157: 00 000 NULL
07158: 00 000 NULL
07159: 00 000 NULL
0715A: 00 000 NULL
0715B: 00 000 NULL
0715C: 00 000 NULL
0715D: 00 000 NULL

```

Variables:

size: byte elements: 8 show as: hex

N1 05h, 07h, 08h, 10h, 14h, 19h, 34h, 98h

Code :

2. Sorting in Descending Order :

DATA SEGMENT

N1 DB 10h,14h,7h,8h,98h,19h,34h,5h

SIZE equ \$ - N1

ENDS

CODE SEGMENT

START:

MOV AX,DATA

MOV DS,AX

MOV CH,SIZE

L1:

LEA SI,N1

MOV CL,SIZE

L2:

MOV AL, [SI]

MOV BL, [SI+1]

CMP AL,BL

JNC DOWN

MOV DL,[SI+1]

XCHG [SI],DL

MOV [SI+1],DL

DOWN:

INC SI

DEC CL

JNZ L2

DEC CH

JNZ L1

CODE ENDS

END START

ret

- Here the elements are sorted in descending order in N1 data segment

Before Sorting -

emulator: poa6.2.exe

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	00	00
BX	00	00
CX	00	36
DX	00	00
CS	0711	
IP	0000	
SS	0710	
SP	0000	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

0711:0000 0711:0000

07110: B8 184 1 MOV AX, 00710h
 07111: 10 016 2 MOV DS, AX
 07112: 07 007 BEEP MOV CH, 08h
 07113: 8E 142 3 MOV SI, 00000h
 07114: D8 216 4 MOV CL, 08h
 07115: B5 181 5 MOV AL, [SI]
 07116: 08 008 BACK MOV BL, [SI] + 01h
 07117: BE 190 6 CMP AL, BL
 07118: 00 000 NULL JNB 01Dh
 07119: 00 000 NULL MOV DL, [SI] + 01h
 0711A: B1 177 7 XCHG [SI], DL
 0711B: 08 008 BACK MOV [SI] + 01h, DL
 0711C: 8A 138 8 INC SI
 0711D: 04 004 9 DEC CL
 0711E: 8A 138 8 JNE 0Ch
 0711F: 5C 092 9 DEC CH
 07120: 01 001 0 JNE 07h
 07121: 3A 058 1 NOP
 07122: C3 195 2 NOP
 07123: 73 115 3 NOP
 07124: 08 008 BACK ...

screen source reset aux vars debug stack flags

variables

size: byte elements: 8

edit show as: hex

N1 10h, 14h, 07h, 08h, 98h, 19h, 34h, 05h

After sorting -

emulator: poa6.2.exe

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	07	05
BX	00	00
CX	00	00
DX	00	19
CS	0711	
IP	003A	
SS	0710	
SP	0000	
BP	0000	
SI	0000	
DI	0000	
DS	0710	
ES	0700	

0711:003A 0711:003A

07147: 90 144 1 NOP
 07148: 90 144 2 NOP
 07149: 90 144 3 NOP
 0714A: F4 244 4 NOP
 0714B: 00 000 NULL NOP
 0714C: 00 000 NULL NOP
 0714D: 00 000 NULL NOP
 0714E: 00 000 NULL NOP
 0714F: 00 000 NULL NOP
 07150: 00 000 NULL NOP
 07151: 00 000 NULL NOP
 07152: 00 000 NULL NOP
 07153: 00 000 NULL NOP
 07154: 00 000 NULL NOP
 07155: 00 000 NULL NOP
 07156: 00 000 NULL NOP
 07157: 00 000 NULL NOP
 07158: 00 000 NULL HLT
 07159: 00 000 NULL ADD [BX + SI], AL
 0715A: 00 000 NULL ADD [BX + SI], AL
 0715B: 00 000 NULL ...

screen source reset aux vars debug stack flags

variables

size: byte elements: 8

edit show as: hex

N1 98h, 34h, 19h, 14h, 10h, 08h, 07h, 05h