

Exercise 6 – Sorting

6A – Sorting in Ascending Order

Aim:

To perform bubble sort in ascending order.

Procedure for executing MASM:

1. Mount the local folder in the DOS-BOX using a temp disk name:
``mount <disk-name> <folder-location>``
2. Change directory into the mounted disk: ``<disk-name>: ``
3. Assemble the instructions: ``masm <file-name>.asm``
4. Link the object file(s) to produce an executable file(.exe): ``link <file-name>.obj; `` Note that removal of semi-colon will make linking process interactive.
5. Debug the executable file to read the memory map and execute the program: ``debug <file-name>.exe``. After entering debug mode,
 - a. ``d <segment:offset> `` - dump(read) memory map from the given location
 - b. ``e <segment:offset> `` - edit memory values from the given location. Use 'White space' to continue editing and 'new line' to exit editing.
 - c. ``u `` - unassemble code (with or without <segment:offset>)
 - d. ``g `` - execute the program
 - e. ``? `` - display command list
 - f. ``q`` - quit the debugger

Algorithm:

1. Initialise data and extra segment using their respective registers.
2. Load CX with count, length of array to sort.
3. Iterate until CX != 0:
 - a. Load DX with CX
 - b. Load SI, DI with two consecutive elements using base address arr
 - c. Compare SI and DI using AL as intermediate.
 - d. If CF = 1, meaning value at SI < value at DI, since consecutive elements already in-place, jump swapping if carry is set.
 - e. Else if meaning value at SI > value at DI, swap using XCHG.
 - f. Increment SI, DI and Decrement DX to run the inner loop
4. Terminate the program.

Program:

Program	Comment
<pre> ; 6a: Sorting in ascending order assume cs: code, ds: data data segment count db 03H arr db 75H, 33H, 01H, 25H data ends code segment start: mov ax, data mov ds, ax mov cx, 00H mov cl, count outer: mov dx, cx mov si, offset arr mov di, offset arr + 1 inner: mov al, [si] cmp al, [di] jc skip xchg al, [di] mov [si], al skip: inc si inc di dec dx jnz inner loop outer mov ah, 4cH int 21H code ends end start </pre>	<p>Comment after ';' </p> <p>Map CS to code segment, DS to data segment</p> <p>Initialise data segment and extra segment db = define a byte Initialise count Initialise arr</p> <p>Initialise code segment Move the starting address of data segment in ax, then move ax to ds.</p> <p>Load CX with count, outer loop itr: n - 1</p> <p>Load DX with CX of each iteration, inner loop itr: n - i - 1 Load SI, DI with two consecutive elements</p> <p>Load AL with value at SI Compare AL against value at DI</p> <p>SI < DI => CF = 1, already in place, so can skip swapping</p> <p>Exchange value at SI with DI using AL as intermediate.</p> <p>Increment SI and DI</p> <p>Decrement DX to run the inner loop, jump if ZF not set.</p> <p>Loop if CX != 0</p> <p>Set ah = 4cH Call interrupt routine 21H for DOS, which terminates if ah = 4cH</p>

Unassembled code:

```

076B:0000 B86A07      MOV     AX,076A
076B:0003 8ED8        MOV     DS,AX
076B:0005 B500        MOV     CH,00
076B:0007 8A0E0000     MOV     CL,[0000]
076B:000B 8BD1        MOV     DX,CX
076B:000D BE0100     MOV     SI,0001
076B:0010 BF0200     MOV     DI,0002
076B:0013 8A04        MOV     AL,[SI]
076B:0015 3A05        CMP     AL,[DI]
076B:0017 7204        JB      001D
076B:0019 8605        XCHG    AL,[DI]
076B:001B 8804        MOV     [SI],AL
076B:001D 46          INC     SI
076B:001E 47          INC     DI
076B:001F 4A          DEC     DX
076B:0020 75F1        JNZ     0013
076B:0022 E2E7        LOOP   000B
076B:0024 B44C        MOV     AH,4C
076B:0026 CD21        INT     21

```

Snapshot of sample input and output:

Before execution:

count = 03H

arr = [75H, 33H, 01H, 25H]

```

-d 076a:0000
076A:0000 03 75 33 01 25 00 00 00-00 00 00 00 00 00 00 00 .u3.%.
076A:0010 B8 6A 07 8E D8 B5 00 8A-0E 00 00 8B D1 BE 01 00 .j.
076A:0020 BF 02 00 8A 04 3A 05 72-04 86 05 88 04 46 47 4A .:r.FGJ
076A:0030 75 F1 E2 E7 B4 4C CD 21-46 18 76 06 89 46 18 89 u...L.!F.v..F..
076A:0040 56 1A B8 04 00 50 0E E8-A6 0A B8 81 27 50 FF 76 U...P.....'P.v
076A:0050 1A FF 76 18 B8 F2 52 50-0E E8 9E 08 83 C4 0A A1 ..v...RP.....
076A:0060 F8 56 0B 06 FA 56 74 1B-B8 08 00 50 0E E8 80 0A .U...Ut....P....
076A:0070 FF 36 FA 56 FF 36 F8 56-B8 FA 52 50 0E E8 7A 08 .6.U.6.U..RP..z.

```

After execution:

count = 03H

arr = [01H, 25H, 33H, 75H]

```

-g
Program terminated normally
-d 076a:0000
076A:0000  03 01 25 33 75 00 00 00-00 00 00 00 00 00 00 00 00  ..%3u.....
076A:0010  B8 6A 07 8E D8 B5 00 8A-0E 00 00 8B D1 BE 01 00  .j.....
076A:0020  BF 02 00 8A 04 3A 05 72-04 86 05 88 04 46 47 4A  .:r....FGJ
076A:0030  75 F1 E2 E7 B4 4C CD 21-46 18 76 06 89 46 18 89  u....L.!F.v..F..
076A:0040  56 1A B8 04 00 50 0E E8-A6 0A B8 81 27 50 FF 76  V....P.....'P.v
076A:0050  1A FF 76 18 B8 F2 52 50-0E E8 9E 08 83 C4 0A A1  ..v...RP.....
076A:0060  F8 56 0B 06 FA 56 74 1B-B8 08 00 50 0E E8 80 0A  .V...Ut....P....
076A:0070  FF 36 FA 56 FF 36 F8 56-B8 FA 52 50 0E E8 7A 08  .6.V.6.V..RP..z.

```

Result:

Program to sort an array in ascending order using bubble sort is assembled, executed and verified.

6B - Sorting in Descending Order

Aim:

To perform bubble sort in descending order.

Algorithm:

1. Initialise data and extra segment using their respective registers.
2. Load CX with count, length of array to sort.
3. Iterate until CX != 0:
 - a. Load DX with CX
 - b. Load SI, DI with two consecutive elements using base address arr
 - c. Compare SI and DI using AL as intermediate.
 - d. If CF = 0, meaning value at SI > value at DI, since consecutive elements already in-place, jump swapping if carry is not set.
 - e. Else if meaning value at SI < value at DI, swap using XCHG.
 - f. Increment SI, DI and Decrement DX to run the inner loop
4. Terminate the program.

Program:

Program	Comment
<pre> ; 6b: Sorting in descending order assume cs: code, ds: data data segment count db 03H arr db 75H, 33H, 01H, 25H data ends code segment start: mov ax, data mov ds, ax mov cx, 00H mov cl, count outer: mov dx, cx mov si, offset arr mov di, offset arr + 1 inner: mov al, [si] cmp al, [di] jnc skip xchg al, [di] mov [si], al skip: inc si inc di dec dx jnz inner loop outer mov ah, 4cH int 21H code ends end start </pre>	<p>Comment after ';' </p> <p>Map CS to code segment, DS to data segment</p> <p>Initialise data segment and extra segment db = define a byte Initialise count Initialise arr</p> <p>Initialise code segment Move the starting address of data segment in ax, then move ax to ds.</p> <p>Load CX with count, outer loop itr: n - 1</p> <p>Load DX with CX of each iteration, inner loop itr: n - i - 1 Load SI, DI with two consecutive elements</p> <p>Load AL with value at SI Compare AL against value at DI</p> <p>SI > DI => CF = 0, already in place, so can skip swapping</p> <p>Exchange value at SI with DI using AL as intermediate.</p> <p>Increment SI and DI</p> <p>Decrement DX to run the inner loop, jump if ZF not set.</p> <p>Loop if CX != 0</p> <p>Set ah = 4cH Call interrupt routine 21H for DOS, which terminates if ah = 4cH</p>

Unassembled code:

```

076B:0000 B86A07      MOV     AX,076A
076B:0003 8ED8        MOV     DS,AX
076B:0005 B500        MOV     CH,00
076B:0007 8A0E0000     MOV     CL,[0000]
076B:000B 8BD1        MOV     DX,CX
076B:000D BE0100     MOV     SI,0001
076B:0010 BF0200     MOV     DI,0002
076B:0013 8A04        MOV     AL,[SI]
076B:0015 3A05        CMP     AL,[DI]
076B:0017 7304        JNB     001D
076B:0019 8605        XCHG    AL,[DI]
076B:001B 8804        MOV     [SI],AL
076B:001D 46          INC     SI
076B:001E 47          INC     DI
076B:001F 4A          DEC     DX
076B:0020 75F1        JNZ     0013
076B:0022 E2E7        LOOP    000B
076B:0024 B44C        MOV     AH,4C
076B:0026 CD21        INT     21

```

Snapshot of sample input and output:

Before execution:

```

count = 03H
arr = [75H, 33H, 01H, 25H]

```

```

-d 076a:0000
076A:0000 03 75 33 01 25 00 00 00-00 00 00 00 00 00 00 00 .u3.%.....
076A:0010 B8 6A 07 8E D8 B5 00 8A-0E 00 00 8B D1 BE 01 00 .j.....
076A:0020 BF 02 00 8A 04 3A 05 73-04 86 05 88 04 46 47 4A .....:s....FGJ
076A:0030 75 F1 E2 E7 B4 4C CD 21-16 3B 46 FE 77 09 89 46 u....L.!.:F.w..F
076A:0040 FE 8A 46 F9 88 46 F8 FE-46 F9 EB C9 8A 5E F8 B7 ..F..F..F....^..
076A:0050 00 8A 87 48 2F D0 D8 73-17 E8 B6 00 8A 5E F8 B7 ...H/..s.....^..
076A:0060 00 8A 87 48 2F D0 D8 73-07 53 B0 01 50 E8 73 01 ...H/..s..S..P..s.
076A:0070 A0 B6 2C 3A 46 F8 74 7E-C7 46 FA 00 00 8A 46 F8 ...,F.t~.F....F.

```

After execution:

count = 03H

arr = [75H, 33H, 25H, 01H]

```
-g
Program terminated normally
-d 076a:0000
076A:0000  03 75 33 25 01 00 00 00-00 00 00 00 00 00 00 00  .u3%.....
076A:0010  B8 6A 07 8E D8 B5 00 8A-0E 00 00 8B D1 BE 01 00  .j.....
076A:0020  BF 02 00 8A 04 3A 05 73-04 86 05 88 04 46 47 4A  ....:s....FGJ
076A:0030  75 F1 E2 E7 B4 4C CD 21-16 3B 46 FE 77 09 89 46  u....L.!.;F.w..F
076A:0040  FE 8A 46 F9 88 46 F8 FE-46 F9 EB C9 8A 5E F8 B7  ..F..F..F....^..
076A:0050  00 8A 87 48 2F D0 D8 73-17 E8 B6 00 8A 5E F8 B7  ...H/..s....^..
076A:0060  00 8A 87 48 2F D0 D8 73-07 53 B0 01 50 E8 73 01  ...H/..s.S..P.s.
076A:0070  A0 B6 2C 3A 46 F8 74 7E-C7 46 FA 00 00 8A 46 F8  .,:F.t~.F....F.
```

Result:

Program to sort an array in descending order using bubble sort is assembled, executed and verified.