Date: 03-04-2023 **Name:** Ashvath S.P **Reg No**: 2162014

COA LAB
Experiment – 8

Write a program in Assembly Language to perform Bubble Sort.

Algorithm:

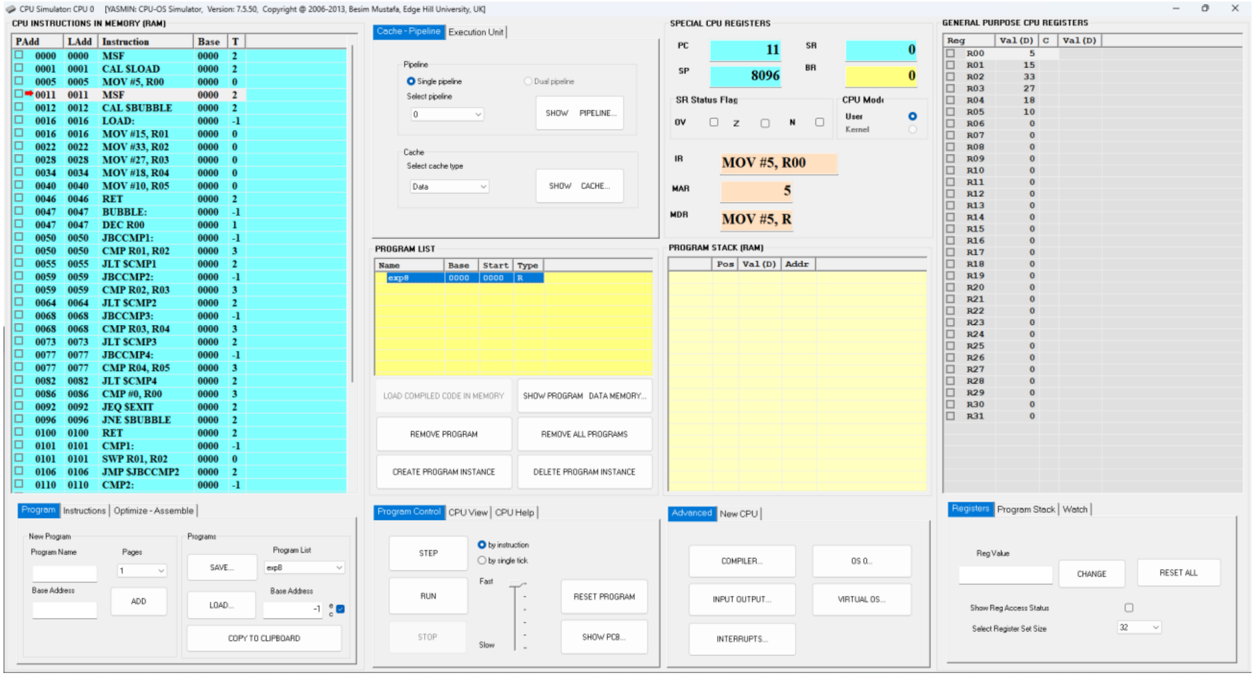
1. Load the first integer into R01, the second into R02, and so on, up to R05.
2. Set R00 to 5, the number of integers in the array.
3. Call the MSF subroutine.
4. In the MSF subroutine, decrement R00 and jump to CMP1 if it is zero.
5. Compare R01 and R02. If R01 is less than R02, jump to CMP1. Otherwise, jump to CMP2.
6. In CMP1, swap R01 and R02 and jump to JBCCMP2.
7. In CMP2, compare R02 and R03. If R02 is less than R03, jump to CMP2. Otherwise, jump to CMP3.
8. In CMP3, swap R03 and R04 and jump to JBCCMP4.
9. In CMP4, compare R04 and R05. If R04 is less than R05, jump to CMP4. Otherwise, jump to BUBBLE.
10. Jump back to the MSF subroutine.
11. When R00 reaches 0, jump to EXIT.
12. The HLT instruction at the end of the program halts execution.

Assembly Language Code:

```
MSF        ; Jump to the main subroutine
CAL $LOAD  ; Call the $LOAD subroutine
MOV #5, R00 ; Move the value 5 to R00, the counter for the bubble sort loop
MSF        ; Call the MSF subroutine to start the bubble sort loop
CAL $BUBBLE ; Call the $BUBBLE subroutine to compare and swap adjacent values
in the array

LOAD       ; Load the array into registers R01-R05
MOV #15, R01 ; Move the value 15 to R01, the first integer in the array
MOV #33, R02 ; Move the value 33 to R02, the second integer in the array
MOV #27, R03 ; Move the value 27 to R03, the third integer in the array
MOV #18, R04 ; Move the value 18 to R04, the fourth integer in the array
MOV #10, R05 ; Move the value 10 to R05, the fifth integer in the array
RET        ; Return from the main subroutine

BUBBLE     ; Start the $BUBBLE subroutine
DEC R00    ; Decrement the counter R00 for the bubble sort loop
JBCCMP1    ; Jump to CMP1 if R00 is zero
```

```
CMP R01, R02 ; Compare R01 and R02
JLT $CMP1  ; Jump to CMP1 if R01 is less than R02
JBCCMP2    ; Jump to CMP2 if R01 is greater than or equal to R02
CMP R02, R03 ; Compare R02 and R03
JLT $CMP2  ; Jump to CMP2 if R02 is less than R03
JBCCMP3    ; Jump to CMP3 if R02 is greater than or equal to R03
CMP R03, R04 ; Compare R03 and R04
JLT $CMP3  ; Jump to CMP3 if R03 is less than R04
JBCCMP4    ; Jump to CMP4 if R03 is greater than or equal to R04
CMP R04, R05 ; Compare R04 and R05
JLT $CMP4  ; Jump to CMP4 if R04 is less than R05
CMP #0, R00 ; Compare R00 to zero
JEQ $EXIT  ; Jump to EXIT if R00 is zero
JNE $BUBBLE ; Jump to BUBBLE if R00 is not zero
RET        ; Return from the $BUBBLE subroutine

CMP1       ; Start the CMP1 subroutine
SWP R01, R02 ; Swap the values in R01 and R02
JMP $JBCCMP2 ; Jump back to JBCCMP2 to continue comparing and swapping adjacent values

CMP2       ; Start the CMP2 subroutine
SWP R02, R03 ; Swap the values in R02 and R03
JMP $JBCCMP3 ; Jump back to JBCCMP3 to continue comparing and swapping adjacent values

CMP3       ; Start the CMP3 subroutine
SWP R03, R04 ; Swap the values in R03 and R04
JMP $JBCCMP4 ; Jump back to JBCCMP4 to continue comparing and swapping adjacent values

CMP4       ; Start the CMP4 subroutine
SWP R04, R05 ; Swap the values in R04 and R05
JMP $BUBBLE ; Jump back to BUBBLE to continue comparing and swapping adjacent values

EXIT       ; Start the EXIT subroutine
HLT        ; Halt the program
```

Result:



CPU Simulator: CPU 0 [YASMIN: CPU-OS Simulator, Version: 7.5.50, Copyright @ 2006-2013, Besim Mustafa, Edge Hill University, UK]

CPU INSTRUCTIONS IN MEMORY (RAM)

PAdd	LAdd	Instruction	Base	T
□ 0000	0000	MSF	0000	2
□ 0001	0001	CAL $LOAD	0000	2
□ 0005	0005	MOV #5, R00	0000	0
□ ➡ 0011	0011	MSF	0000	2
□ 0012	0012	CAL $BUBBLE	0000	2
□ 0016	0016	LOAD:	0000	-1
□ 0016	0016	MOV #15, R01	0000	0
□ 0022	0022	MOV #33, R02	0000	0
□ 0028	0028	MOV #27, R03	0000	0
□ 0034	0034	MOV #18, R04	0000	0
□ 0040	0040	MOV #10, R05	0000	0
□ 0046	0046	RET	0000	2
□ 0047	0047	BUBBLE:	0000	-1
□ 0047	0047	DEC R00	0000	1
□ 0050	0050	JBCCMP1:	0000	-1
□ 0050	0050	CMP R01, R02	0000	3
□ 0055	0055	JLT $CMP1	0000	2
□ 0059	0059	JBCCMP2:	0000	-1
□ 0059	0059	CMP R02, R03	0000	3
□ 0064	0064	JLT $CMP2	0000	2
□ 0068	0068	JBCCMP3:	0000	-1
□ 0068	0068	CMP R03, R04	0000	3
□ 0073	0073	JLT $CMP3	0000	2
□ 0077	0077	JBCCMP4:	0000	-1
□ 0077	0077	CMP R04, R05	0000	3
□ 0082	0082	JLT $CMP4	0000	2
□ 0086	0086	CMP #0, R00	0000	3
□ 0092	0092	JEQ $EXIT	0000	2
□ 0096	0096	JNE $BUBBLE	0000	2
□ 0100	0100	RET	0000	2
□ 0101	0101	CMP1:	0000	-1
□ 0101	0101	SWP R01, R02	0000	0
□ 0106	0106	JMP $JBCCMP2	0000	2
□ 0110	0110	CMP2:	0000	-1

SPECIAL CPU REGISTERS

PC 11 SR 0

SP 8096 BR 0

SR Status Flag: OV □ Z □ N □

CPU Mode: User ● Kernel ○

IR MOV #5, R00

MAR 5

MDR MOV #5, R

PROGRAM LIST

Name	Base	Start	Type
exp8	0000	0000	R

PROGRAM STACK (RAM)

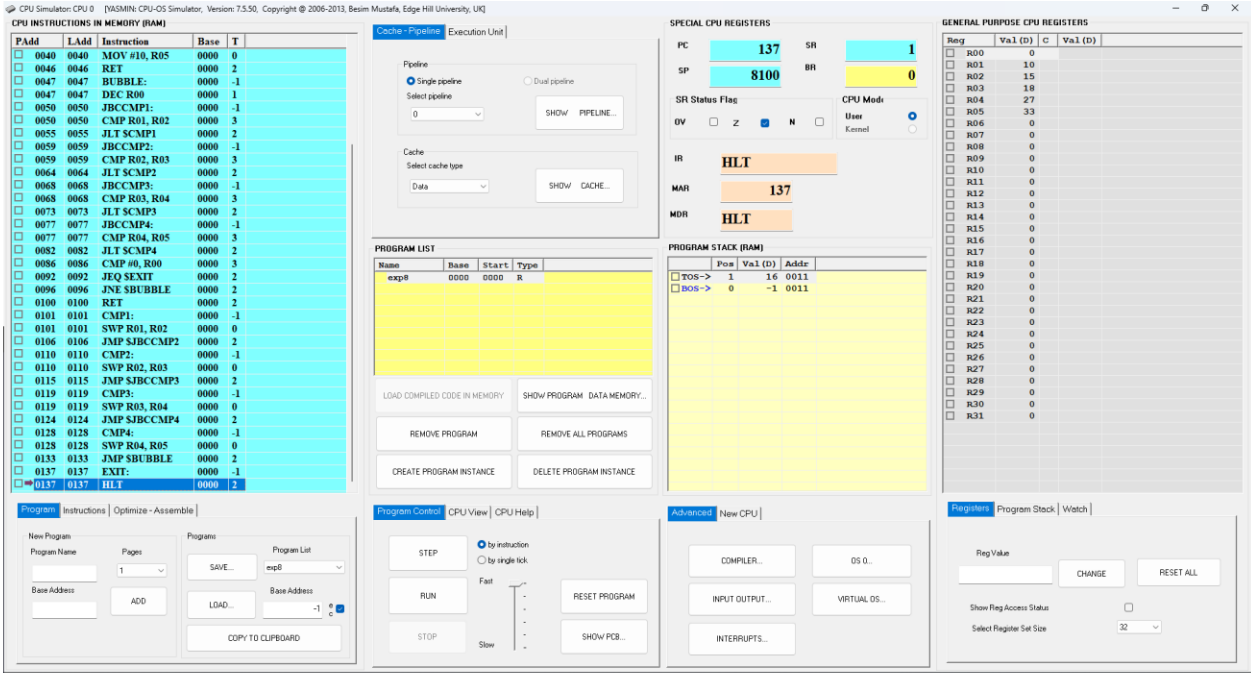
	Pos	Val (D)	Addr

GENERAL PURPOSE CPU REGISTERS

Reg	Val (D)	C	Val (D)
□ R00	5		
□ R01	15		
□ R02	33		
□ R03	27		
□ R04	18		
□ R05	10		
□ R06	0		
□ R07	0		
□ R08	0		
□ R09	0		
□ R10	0		
□ R11	0		
□ R12	0		
□ R13	0		
□ R14	0		
□ R15	0		
□ R16	0		
□ R17	0		
□ R18	0		
□ R19	0		
□ R20	0		
□ R21	0		
□ R22	0		
□ R23	0		
□ R24	0		
□ R25	0		
□ R26	0		
□ R27	0		
□ R28	0		
□ R29	0		
□ R30	0		
□ R31	0		

Program List: exp8

Fig: CPU Simulator Window (Before Sorting)



CPU Simulator: CPU 0 [YASMIN: CPU-OS Simulator, Version: 7.5.50, Copyright @ 2006-2013, Besim Mustafa, Edge Hill University, UK]

CPU INSTRUCTIONS IN MEMORY (RAM)

PAdd	LAdd	Instruction	Base	T
□ 0040	0040	MOV #10, R05	0000	0
□ 0046	0046	RET	0000	2
□ 0047	0047	BUBBLE:	0000	-1
□ 0047	0047	DEC R00	0000	1
□ 0050	0050	JBCCMP1:	0000	-1
□ 0050	0050	CMP R01, R02	0000	3
□ 0055	0055	JLT $CMP1	0000	2
□ 0059	0059	JBCCMP2:	0000	-1
□ 0059	0059	CMP R02, R03	0000	3
□ 0064	0064	JLT $CMP2	0000	2
□ 0068	0068	JBCCMP3:	0000	-1
□ 0068	0068	CMP R03, R04	0000	3
□ 0073	0073	JLT $CMP3	0000	2
□ 0077	0077	JBCCMP4:	0000	-1
□ 0077	0077	CMP R04, R05	0000	3
□ 0082	0082	JLT $CMP4	0000	2
□ 0086	0086	CMP #0, R00	0000	3
□ 0092	0092	JEQ $EXIT	0000	2
□ 0096	0096	JNE $BUBBLE	0000	2
□ 0100	0100	RET	0000	2
□ 0101	0101	CMP1:	0000	-1
□ 0101	0101	SWP R01, R02	0000	0
□ 0106	0106	JMP $JBCCMP2	0000	2
□ 0110	0110	CMP2:	0000	-1
□ 0110	0110	SWP R02, R03	0000	0
□ 0115	0115	JMP $JBCCMP3	0000	2
□ 0119	0119	CMP3:	0000	-1
□ 0119	0119	SWP R03, R04	0000	0
□ 0124	0124	JMP $JBCCMP4	0000	2
□ 0128	0128	CMP4:	0000	-1
□ 0128	0128	SWP R04, R05	0000	0
□ 0133	0133	JMP $BUBBLE	0000	2
□ 0137	0137	EXIT:	0000	-1
□ ➡ 0137	0137	HLT	0000	2

SPECIAL CPU REGISTERS

PC 137 SR 1

SP 8100 BR 0

SR Status Flag: OV □ Z ☑ N □

CPU Mode: User ● Kernel ○

IR HLT

MAR 137

MDR HLT

PROGRAM LIST

Name	Base	Start	Type
exp8	0000	0000	R

PROGRAM STACK (RAM)

	Pos	Val (D)	Addr
□ TOS->	1	16	0011
□ BOS->	0	-1	0011

GENERAL PURPOSE CPU REGISTERS

Reg	Val (D)	C	Val (D)
□ R00	0		
□ R01	10		
□ R02	15		
□ R03	18		
□ R04	27		
□ R05	33		
□ R06	0		
□ R07	0		
□ R08	0		
□ R09	0		
□ R10	0		
□ R11	0		
□ R12	0		
□ R13	0		
□ R14	0		
□ R15	0		
□ R16	0		
□ R17	0		
□ R18	0		
□ R19	0		
□ R20	0		
□ R21	0		
□ R22	0		
□ R23	0		
□ R24	0		
□ R25	0		
□ R26	0		
□ R27	0		
□ R28	0		
□ R29	0		
□ R30	0		
□ R31	0		

Program List: exp8

Fig: CPU Simulator Window (After Sorting)