**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*

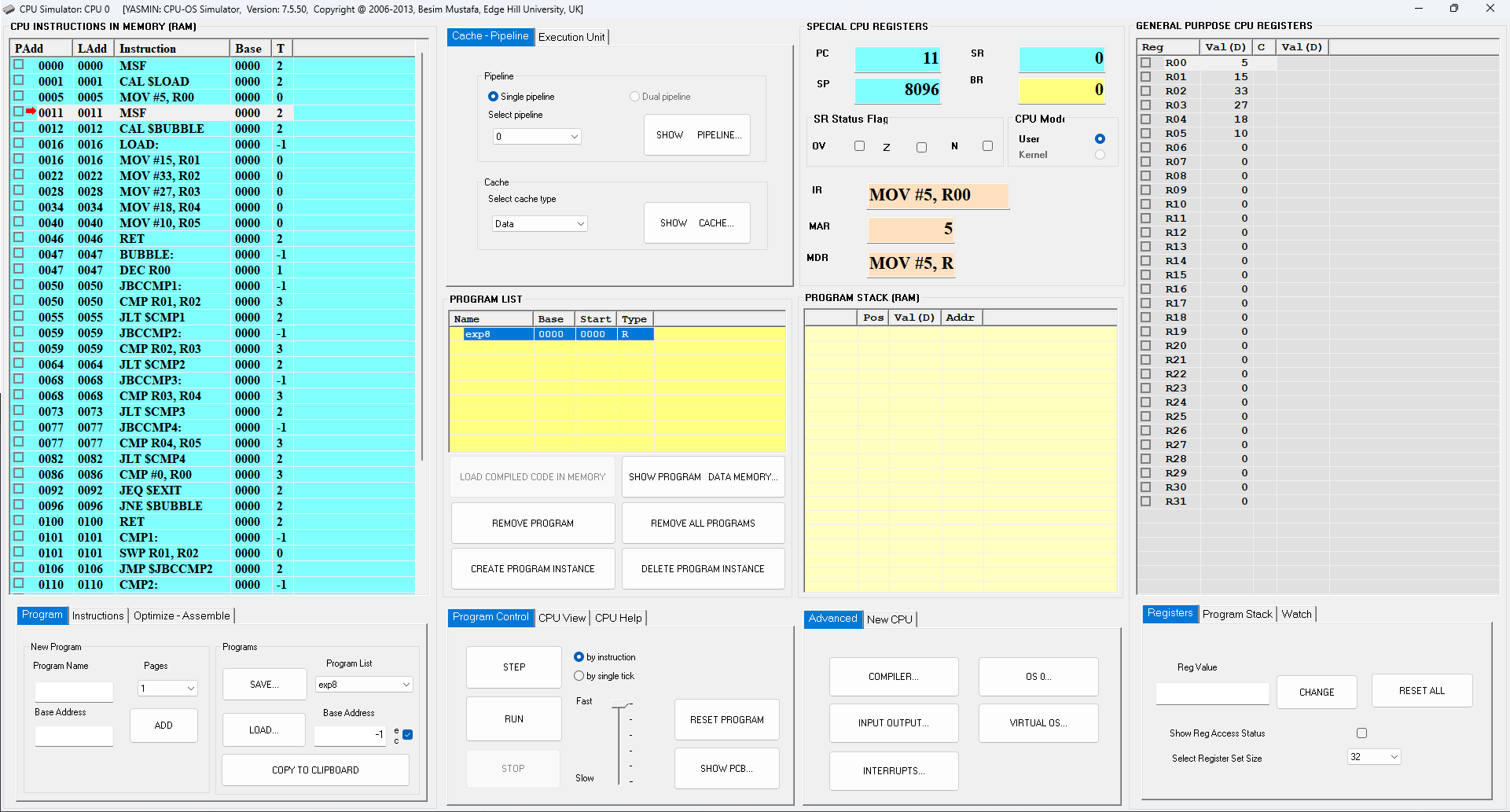
SWP R04, R05 ; *Swap the values in R04 and R05*

JMP $BUBBLE ; *Jump back to BUBBLE to continue comparing and swapping adjacent values*

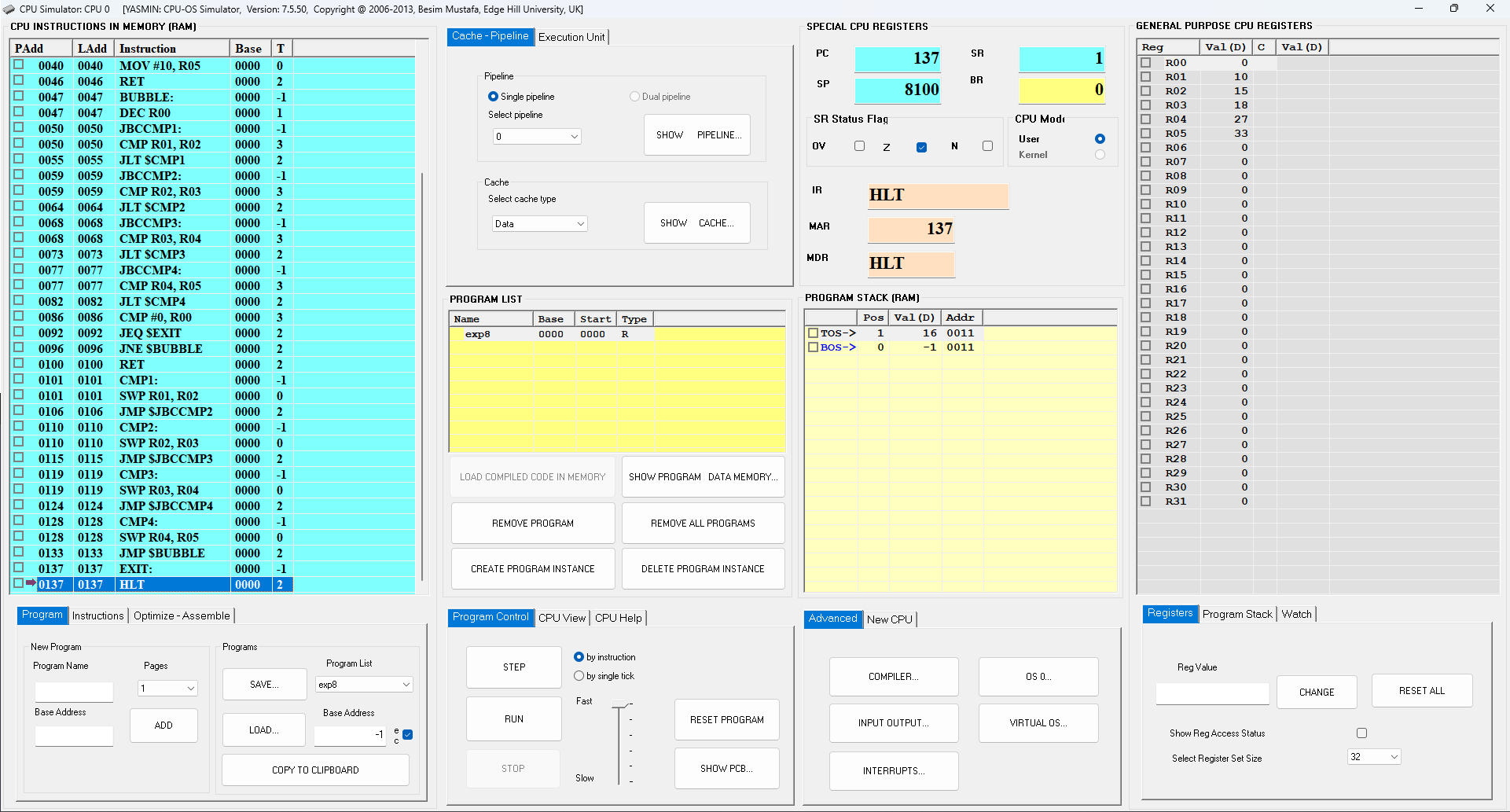
EXIT *; Start the EXIT subroutine*

HLT ; *Halt the program*

**Result:**

****

**Fig: CPU Simulator Window (Before Sorting)**

****

**Fig: CPU Simulator Window (After Sorting)**