# Laboratory 6

Title of the Laboratory Exercise: Sorting

1. Introduction and Purpose of Experiment

Students will create assembly code with sorting techniques and nested loops

1. Aim and Objectives

Aim

To develop assembly language program to perform sorting using nested loop structures

Objectives

At the end of this lab, the student will be able to

* + use nested loops in assembly
  + perform sorting in ascending/ descending order
  + Build complex looping logic in assembly language

1. Experimental Procedure

1. Write algorithm to solve the given problem

2. Translate the algorithm to assembly language code

3. Run the assembly code in GNU assembler

4. Create laboratory report documenting the work

1. Questions

Develop an assembly language program to perform the following

1. To design calculator to perform all arithmetic operations based on input given by user.
2. To perform SWAP operation using Logical instructions
3. To compute factorial of a number.
4. To find second smallest number in an unsorted array.
5. Calculations/Computations/Algorithms

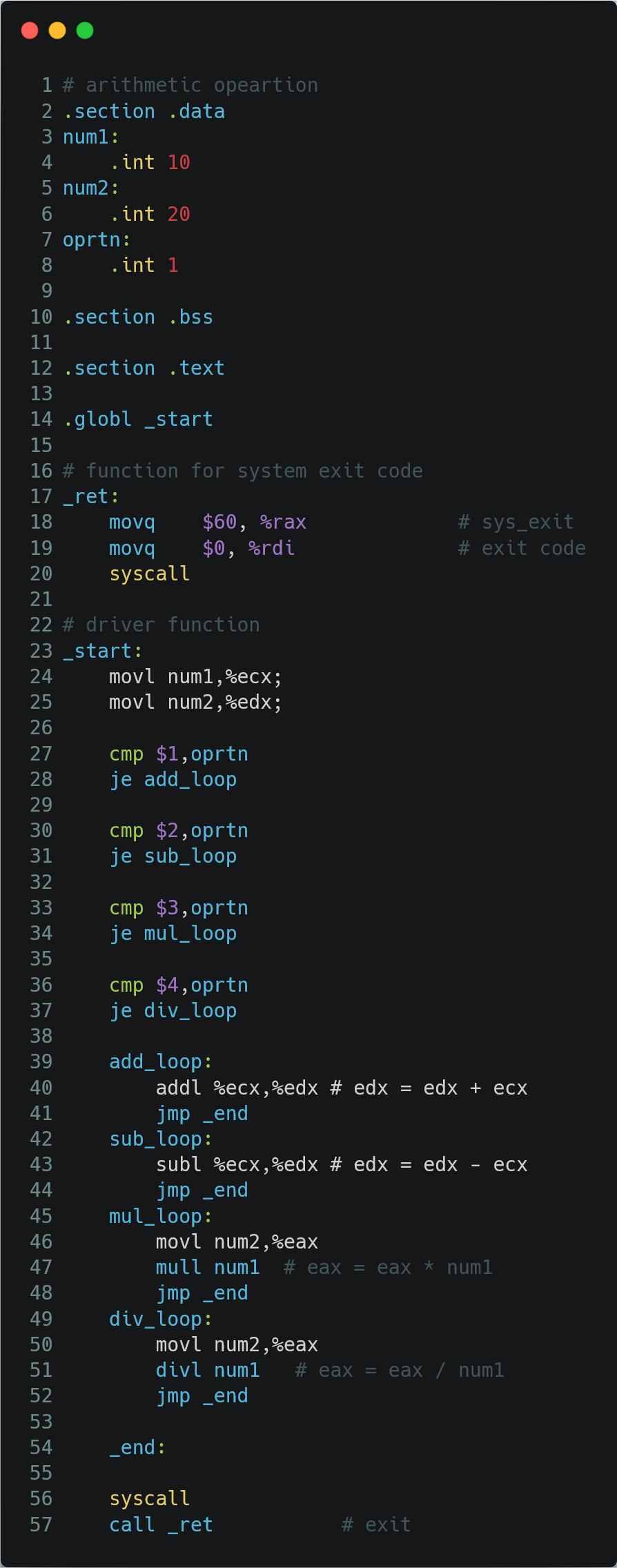


Figure 1 ASM code to design calculator to perform all arithmetic operations based on input given by user.



Figure 2 ASM code to perform SWAP operation using Logical instructions

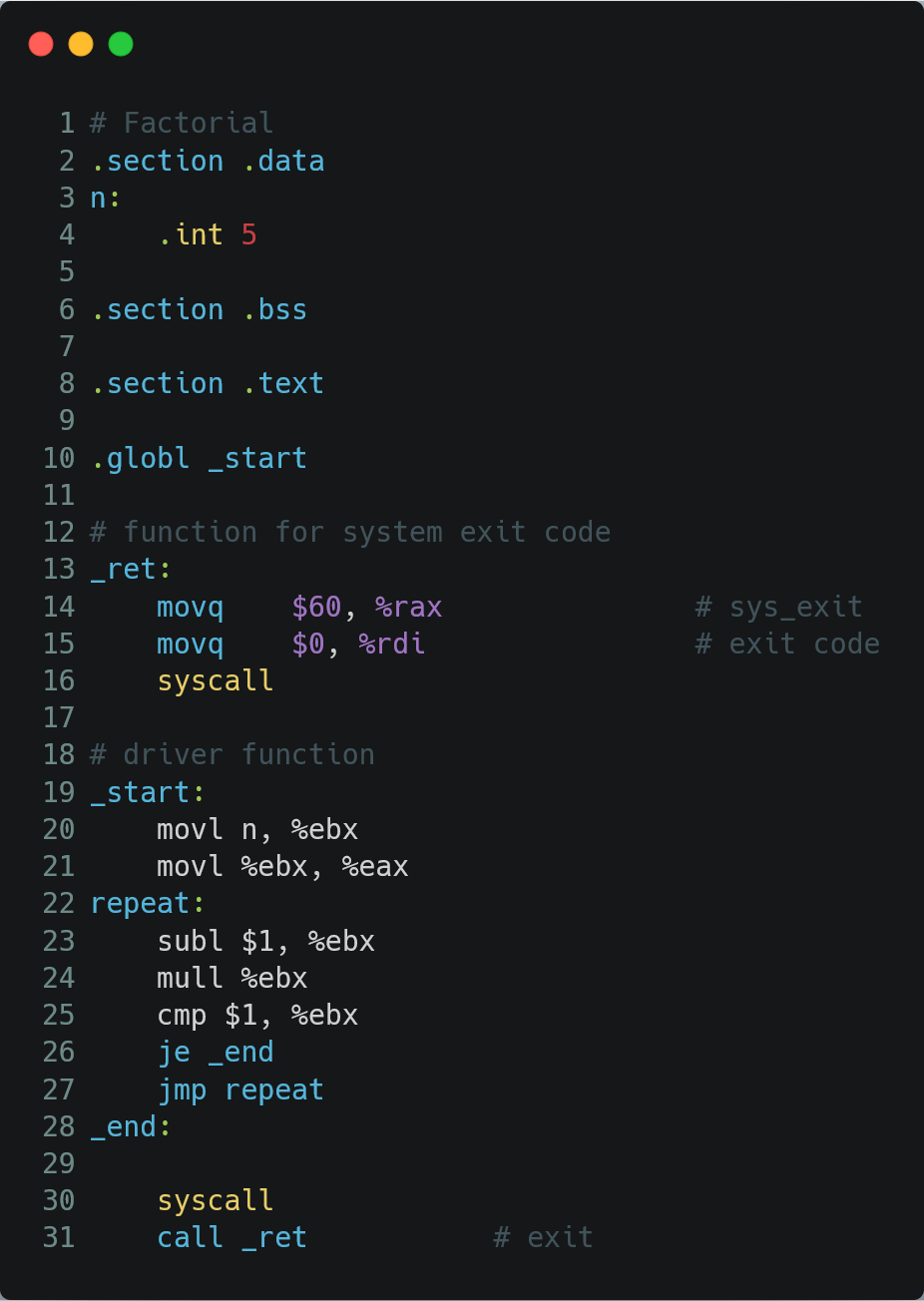


Figure 3 ASM code to compute factorial of a number.

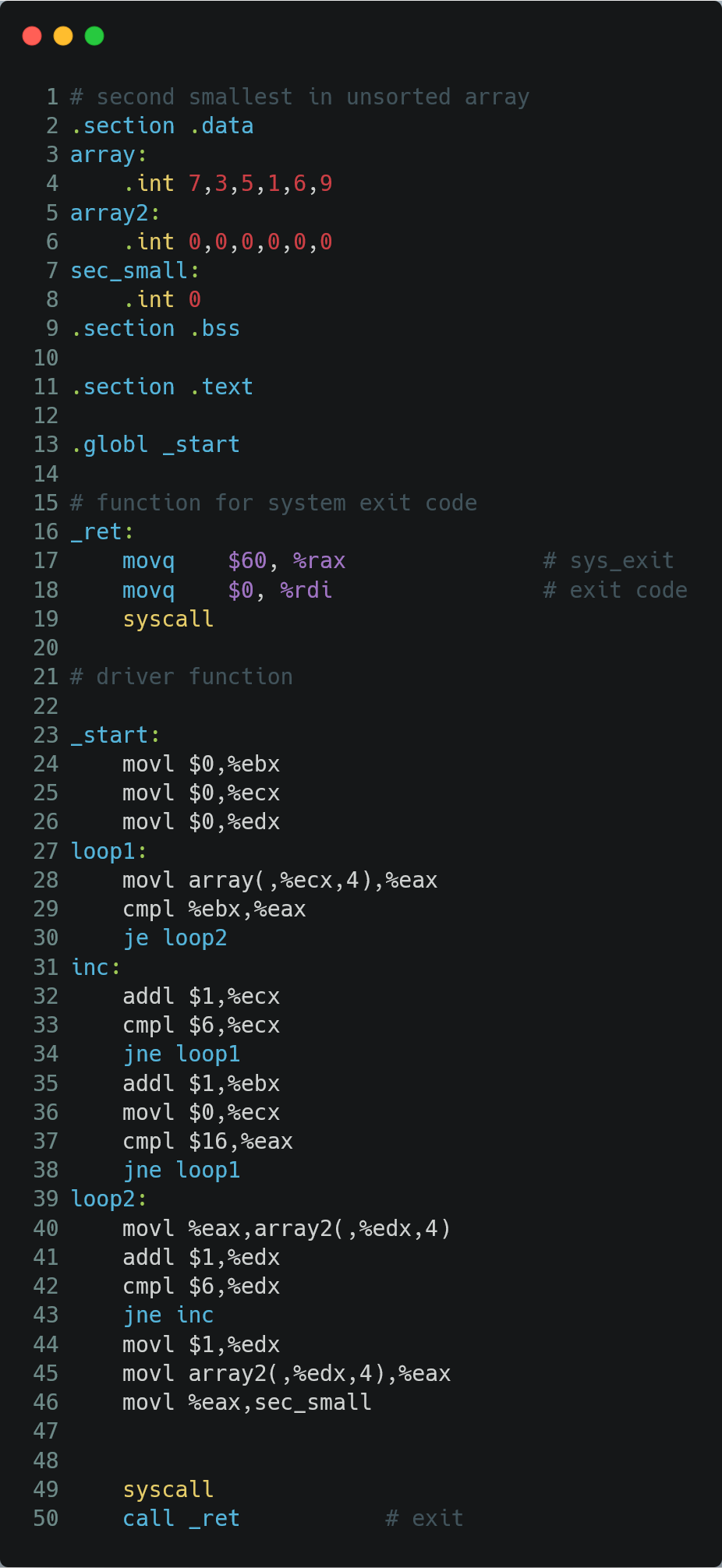


Figure 4 ASM code to find second smallest number in an unsorted array.

1. Presentation of Results

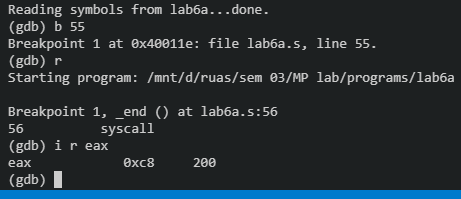


Figure 5 output to design a calculator to perform all arithmetic operations based on input given by user.

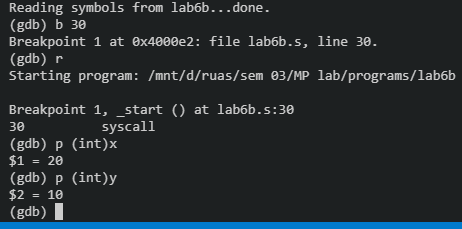


Figure 6 output of program to perform SWAP operation using Logical instructions

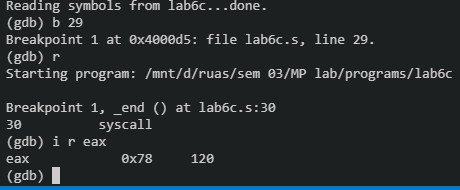


Figure 7 output of code to compute factorial of a number

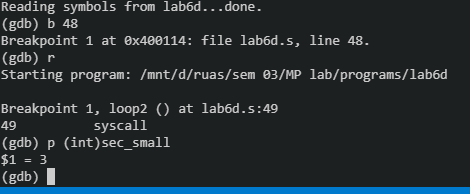


Figure 8 output of program to find second smallest number in an unsorted array.

1. Analysis and Discussions

|  |  |
| --- | --- |
| Code | jcc address |
| Example | jne loop |
| Explanation | Performs:  Jumps to the address location if the condition is met  Here cc = ne, e, ge, g, etc.  Description:  Checks the state of one or more of the status flags in the EFLAGS register (CF, OF, PF, SF, and ZF) and, if the flags are in the specified state (condition), performs a jump to the target instruction specified by the destination operand. A condition code (cc) is associated with each instruction to indicate the condition being tested for. If the condition is not satisfied, the jump is not performed and execution continues with the instruction following the Jcc instruction. |

|  |  |
| --- | --- |
| Code | cmp op1 op2 |
| Example | cmp $0, %eax |
| Explanation | Performs:  Compares the two operands  Description:  Compares the first source operand with the second source operand and sets the status flags in the EFLAGS register according to the results. The comparison is performed by subtracting the second operand from the first operand and then setting the status flags in the same manner as the SUB instruction. When an immediate value is used as an operand, it is sign-extended to the length of the first operand. |

|  |  |
| --- | --- |
| Code | xor <source> <destination> |
| Example | xorl $20, %ebx |
| Explanation | Performs:  Destination = Destination XOR Source  Description:  Performs a bitwise exclusive OR (XOR) operation on the destination (first) and source (second) operands and stores the result in the destination operand location. The source operand can be an immediate, a register, or a memory location; the destination operand can be a register or a memory location. Each bit of the result is 1 if the corresponding bits of the operands are different; each bit is 0 if the corresponding bits are the same. |

1. Conclusions

In the conclusion we have learnt about the different sorting techniques and nested loops using the assembly language program for performing different operation using compare instruction and flag conditions to perform logical operations.

1. Comments
   1. Limitations of Experiments

Every time we must declare which input from the user has to be executed to perform any arithmetic operation in calculator

* 1. Limitations of Results

None.

* 1. Learning happened

We have learnt to develop assembly language program to perform sorting using nested loop structures and perform sorting in ascending/ descending order and build complex looping logic in assembly language

Signature and date