

# SELECTION SORT ALGORITHM

TEAM MEMBERS:

NIHARKA -PES1UG20EC315

POORVI RADDI-PES1UG20EC318

# PROBLEM STATEMENT

To apply the selection sort algorithm to sort (in ascending order) a given array using assembly code

# WHAT IS SELECTION SORT ALGORITHM ?

The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from the unsorted part and putting it at the beginning.

The algorithm maintains two subarrays in a given array.

- The subarray which already sorted.
- The remaining subarray was unsorted.



**AN  
EXAMPLE**

# METHODOLOGY

- Initialize minimum value(min\_id) to location 0.
- Traverse the array to find the minimum element in the array.
- While traversing if any element smaller than min\_id is found then swap both the values.
- Then, increment min\_id to point to the next element.
- Repeat until the array is sorted.

# CODE SNIPPETS

```
1 .data
2 arr: .word 1, 5, 4, 3, 10
3
4 .text
5 main:
6     #get array base address
7     la x3, arr
8     #init i
9     addi x5, x0, 0
10 #for loop i
11 L1:
12     #init min_id = i
13     addi x7, x5, 0
14     #init j = i + 1
15     addi x6, x5, 1
```

- Define elements of an array

## Main module

- Loading base address of the array
- Initializing i as 0

## Loop for i

- set min\_id as i
- set j= i+1

```

#for loop j
L2:
    #get arr[j] address offset and address
    slli x29, x6, 2
    add x13, x3, x29
    lw x10, 0(x13)
    #get arr[min_id] address offset and address
    slli x30, x7, 2
    add x14, x3, x30
    lw x11, 0(x14)
    #check if condition x10 = arr[j] ; x11 = arr[min_id]
    bge x10, x11, nochange
    addi x7, x6, 0

```

## Loop for j

- Finding the address of arr[j] and arr[min\_id]
- checks if arr[j] is greater than arr[min\_id]

```

29 nochange:
30     #j++
31     addi x6, x6, 1
32     addi x19, x0, 5
33     blt x6, x19, L2
34     #get arr[i] address offset and address
35     slli x28, x5, 2
36     add x12, x3, x28
37     #get arr[min_id] address offset and address
38     slli x30, x7, 2
39     add x14, x3, x30
40     # temp = arr[i]
41     lw x10, 0(x12)
42     # arr[i] = arr[min_id]
43     lw x11, 0(x14)
44     sw x11, 0(x12)
45     # arr[min_id] = temp
46     sw x10, 0(x14)
47     #i++
48     addi x5, x5, 1
49     addi x18, x0, 4
50     blt x5, x18, L1

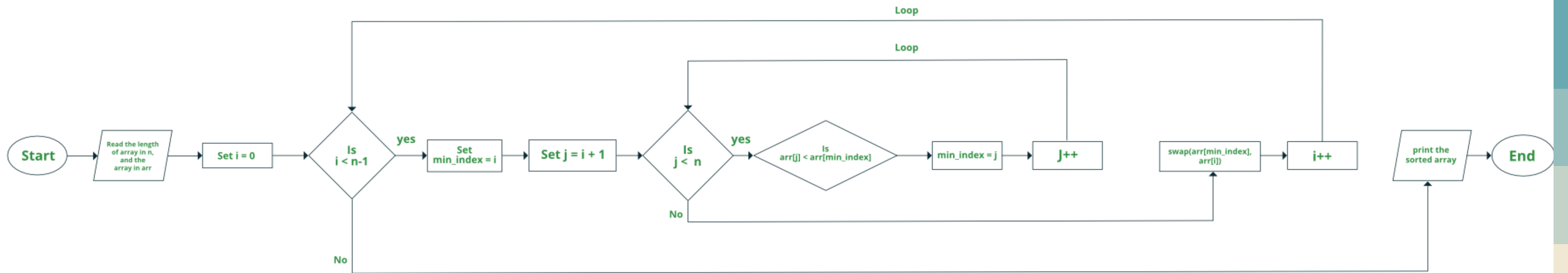
```

## No-Change Loop:

- j increments by 1
- x19 register holds the value of the length of the array.
- checks the condition if j is less than the value in x19 and then increments j
- finds the address of arr[i] and arr[min\_id] and swapping function is performed.
- if arr[min\_id] is less than arr[i]
- then i is incremented by 1



# FLOWCHART



The image shows a screenshot of the Ripes IDE interface. The main window is titled "Ripes" and has a menu bar with "File", "Edit", "View", and "Help". Below the menu bar is a toolbar with icons for running, stepping through code, and a time delay of 100 ms. On the left side, there is a vertical toolbar with icons for "100 1010 01 Editor", "Processor", "Cache", "Memory", and "I/O". The central area is divided into two panes. The left pane is titled "Memory viewer" and displays a table of memory addresses and their corresponding word and byte values. The right pane is titled "Memory map" and displays a table of memory segments with their names, sizes, and ranges. At the bottom of the window, there is a status bar showing "Processor: 5-stage processor" and "ISA: RV32IMC".

The image shows a screenshot of the Ripes IDE interface. The main window is titled "Ripes" and has a menu bar with "File", "Edit", "View", and "Help". Below the menu bar is a toolbar with icons for running, stepping through code, and a time delay of 100 ms. On the left side, there is a vertical toolbar with icons for "100 1010 01 Editor", "Processor", "Cache", "Memory", and "I/O". The main area is divided into two panes. The left pane is titled "Memory viewer" and displays a table of memory addresses and their corresponding data. The right pane is titled "Memory map" and displays a table of memory segments. The status bar at the bottom indicates "Processor: 5-stage processor" and "ISA: RV32IMC".

The image shows a screenshot of the Ripes debugger interface. The main window is titled "Ripes" and has a menu bar with "File", "Edit", "View", and "Help". Below the menu bar is a toolbar with icons for running, stepping, and other debugging actions, along with a "100 ms" timer. On the left side, there is a vertical sidebar with icons for "Editor", "Processor", "Cache", "Memory", and "I/O". The "Memory" icon is currently selected. The main area is divided into two panes. The left pane is titled "Memory viewer" and displays a table of memory addresses and their corresponding data. The right pane is titled "Memory map" and displays a table of memory segments. The "Memory viewer" table has columns for "Address", "Word", "Byte 0", "Byte 1", "Byte 2", and "Byte 3". The "Memory map" table has columns for "Name", "Size", and "Range". The "Memory viewer" table shows a range of addresses from 0x1000002c to 0xfffffd8, with data values ranging from 0 to 10. The "Memory map" table shows three segments: ".text" (Size 108, Range 0x00000000 - 0x0000006c), ".data" (Size 20, Range 0x10000000 - 0x10000014), and ".bss" (Size 0, Range 0x11000000 - 0x11000000). At the bottom of the "Memory viewer" pane, there are controls for "Display type" (set to "Signed"), "Go to register" (set to "Select"), and "Go to section" (set to an empty dropdown).

100  
1010  
01  
Editor

Processor

Cache

Memory

I/O

File Edit View Help

100 ms

Memory viewer

Address	Word	Byte 0	Byte 1	Byte 2	Byte 3
0x1000002c	X	X	X	X	X
0x10000028	X	X	X	X	X
0x10000024	X	X	X	X	X
0x10000020	X	X	X	X	X
0x1000001c	X	X	X	X	X
0x10000018	X	X	X	X	X
0x10000014	X	X	X	X	X
0x10000010	10	10	0	0	0
0x1000000c	5	5	0	0	0
0x10000008	4	4	0	0	0
0x10000004	3	3	0	0	0
0x10000000	1	1	0	0	0
0xffffffffc	X	X	X	X	X
0xffffffff8	X	X	X	X	X
0xffffffff4	X	X	X	X	X
0xffffffff0	X	X	X	X	X
0xfffffffec	X	X	X	X	X
0xffffffe8	X	X	X	X	X
0xffffffe4	X	X	X	X	X
0xffffffe0	X	X	X	X	X
0xffffffd8	X	X	X	X	X

Memory map

Name	Size	Range
.text	108	0x00000000 - 0x0000006c
.data	20	0x10000000 - 0x10000014
.bss	0	0x11000000 - 0x11000000

Display type: Signed

Go to register: Select

Go to section:

**THANK YOU**