RISC-V ARCHITECTURE PROJECT

SELECTION SORT ALGORITHM

TEAM MEMBERS:
NIHARKA -PES1UG20EC315
POORVI RADDI-PES1UG20EC318

PROBLEM STATEMENT

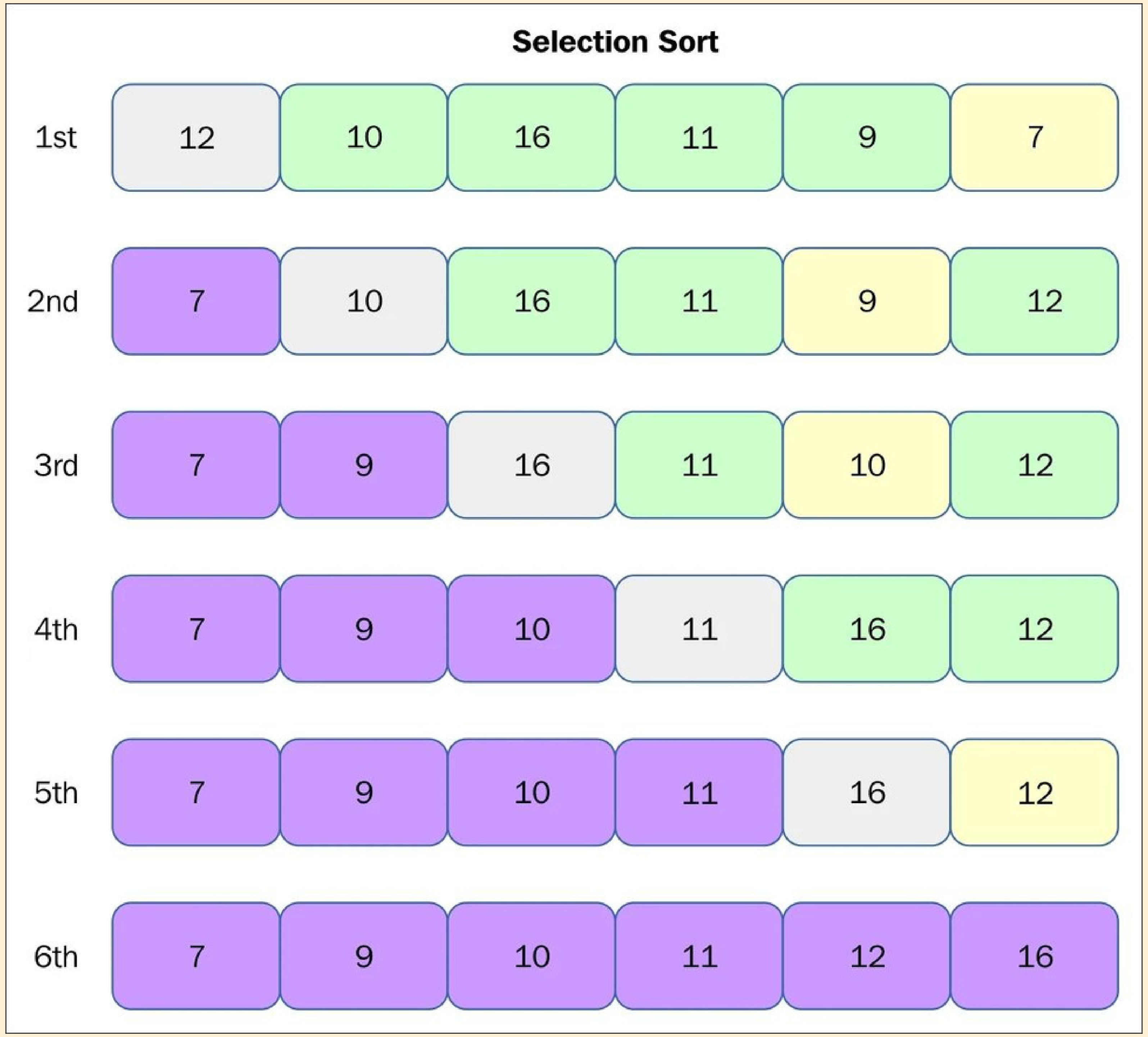
To apply the selection sort algorithm to sort
(in ascending order) a given array using assembly code

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



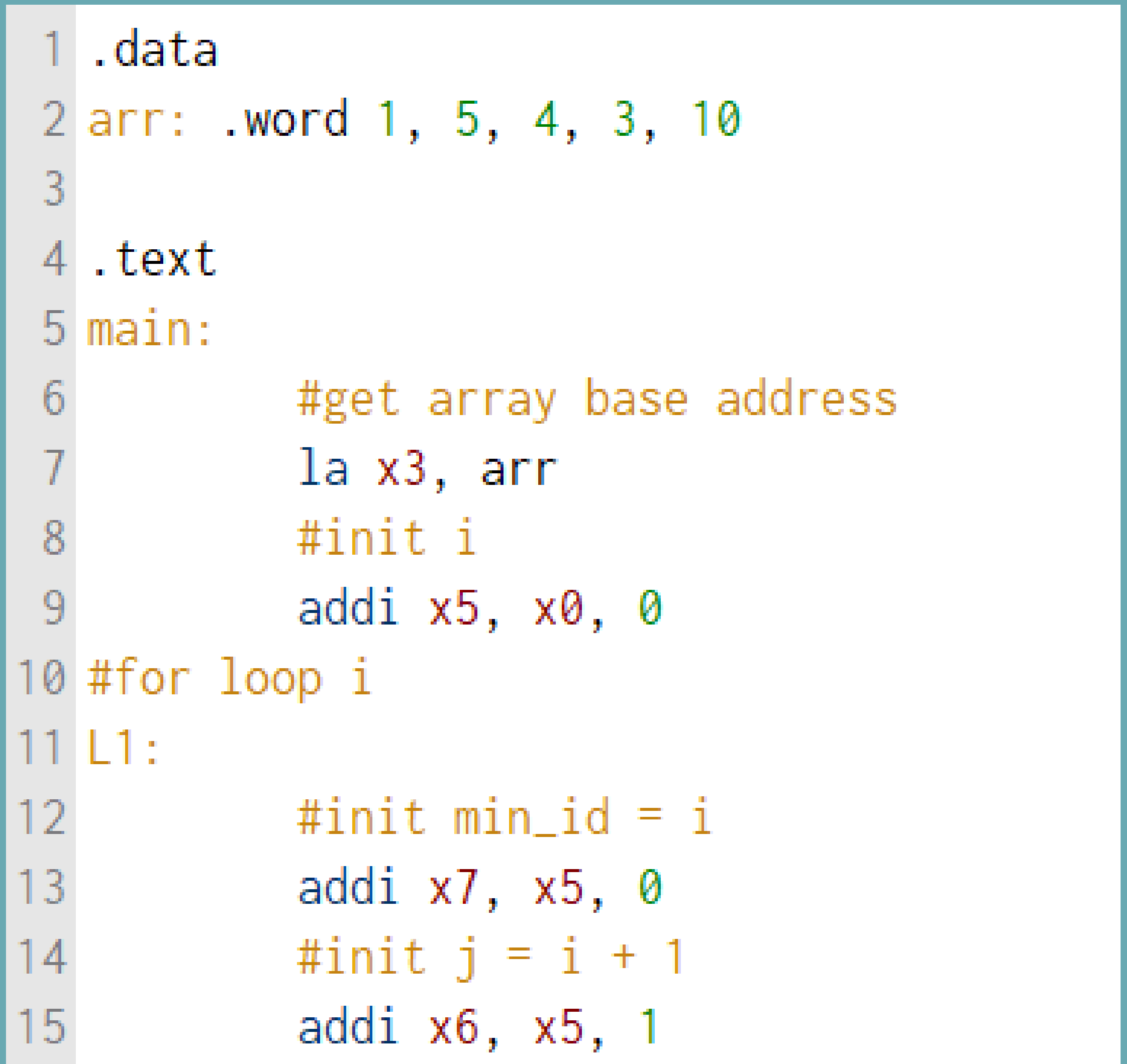
Selection Sort

1st	12	10	16	11	9	7
2nd	7	10	16	11	9	12
3rd	7	9	16	11	10	12
4th	7	9	10	11	16	12
5th	7	9	10	11	16	12
6th	7	9	10	11	12	16

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



```
.data
arr: .word 1, 5, 4, 3, 10

.text
main:
        #get array base address
        la x3, arr
        #init i
        addi x5, x0, 0
#for loop i
L1:
        #init min_id = i
        addi x7, x5, 0
        #init j = i + 1
        addi x6, x5, 1
```

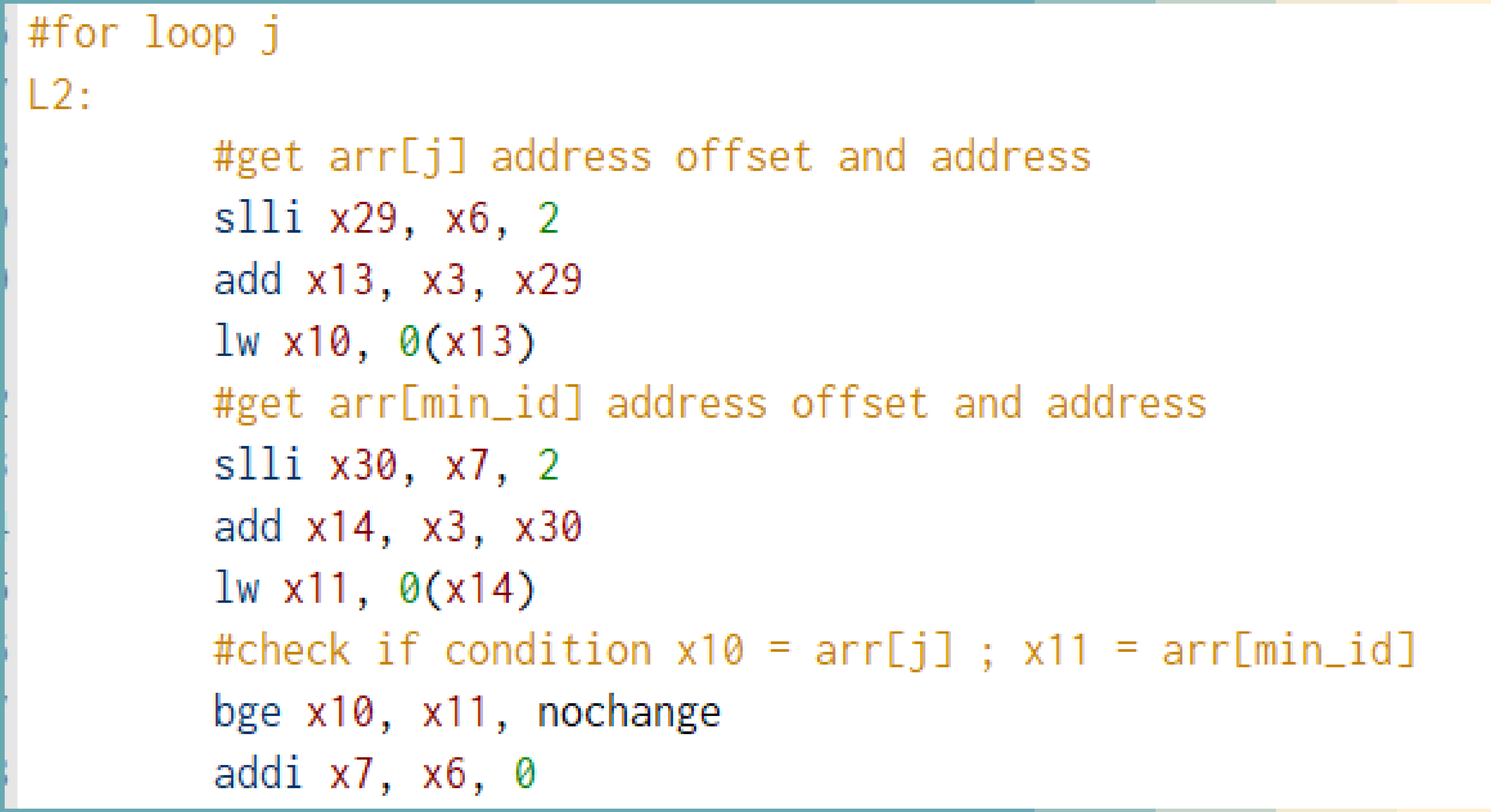
- Define elements of an array

Main module

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

Loop for i

- set min_id as i
- set j= i+1

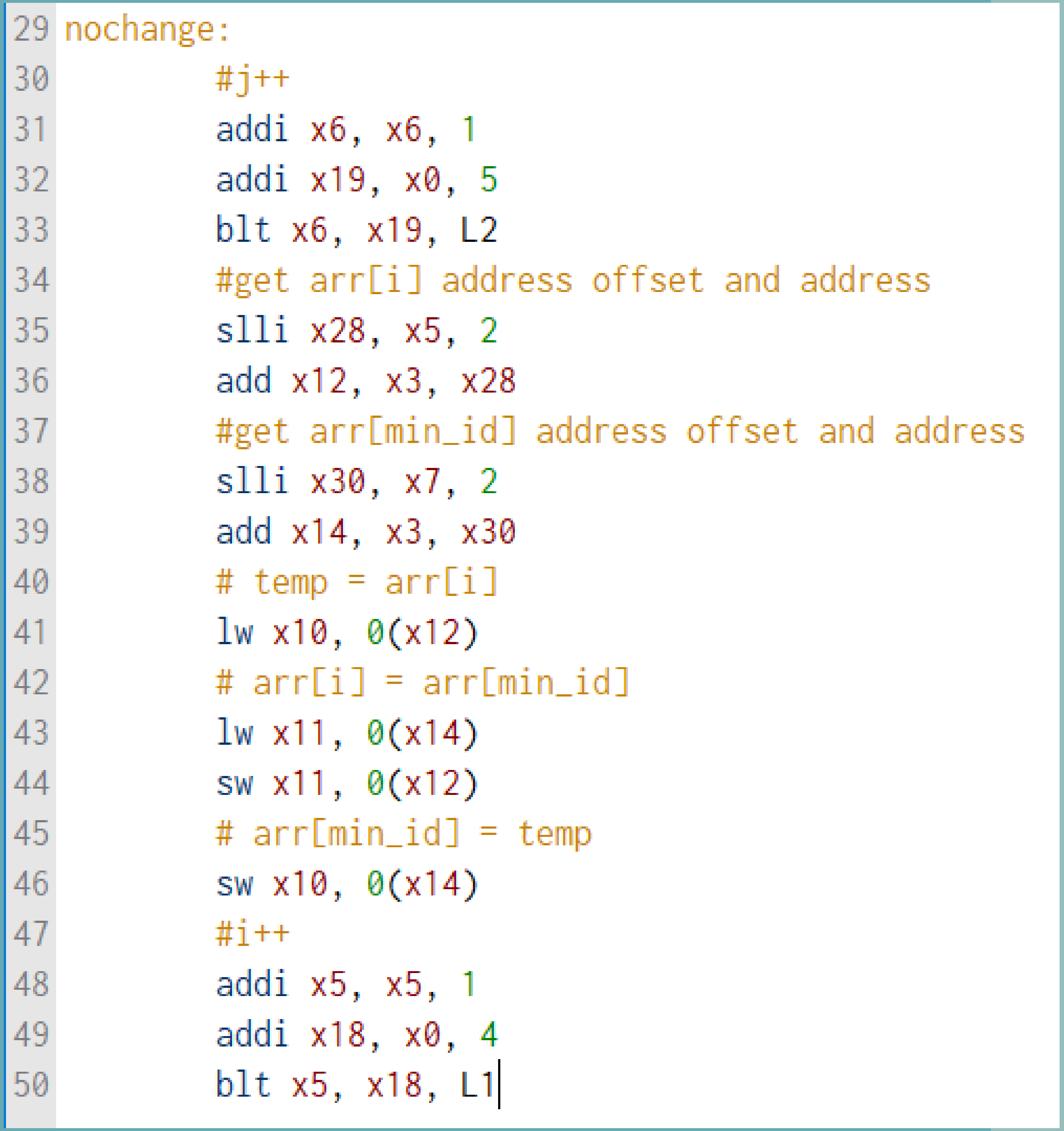


```
#for loop j
L2:
        #get arr[j] address offset and address
        slli x29, x6, 2
        add x13, x3, x29
        lw x10, 0(x13)
        #get arr[min_id] address offset and address
        slli x30, x7, 2
        add x14, x3, x30
        lw x11, 0(x14)
        #check if condition x10 = arr[j]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

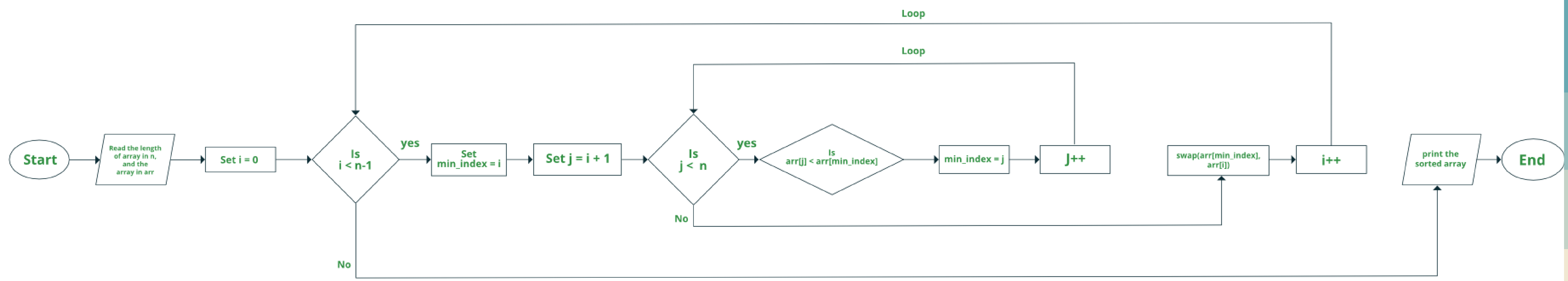
No-Change Loop:



```
nochange:
        #j++
        addi x6, x6, 1
        addi x19, x0, 5
        blt x6, x19, L2
        #get arr[i] address offset and address
        slli x28, x5, 2
        add x12, x3, x28
        #get arr[min_id] address offset and address
        slli x30, x7, 2
        add x14, x3, x30
        # temp = arr[i]
        lw x10, 0(x12)
        # arr[i] = arr[min_id]
        lw x11, 0(x14)
        sw x11, 0(x12)
        # arr[min_id] = temp
        sw x10, 0(x14)
        #i++
        addi x5, x5, 1
        addi x18, x0, 4
        blt x5, x18, L1
```

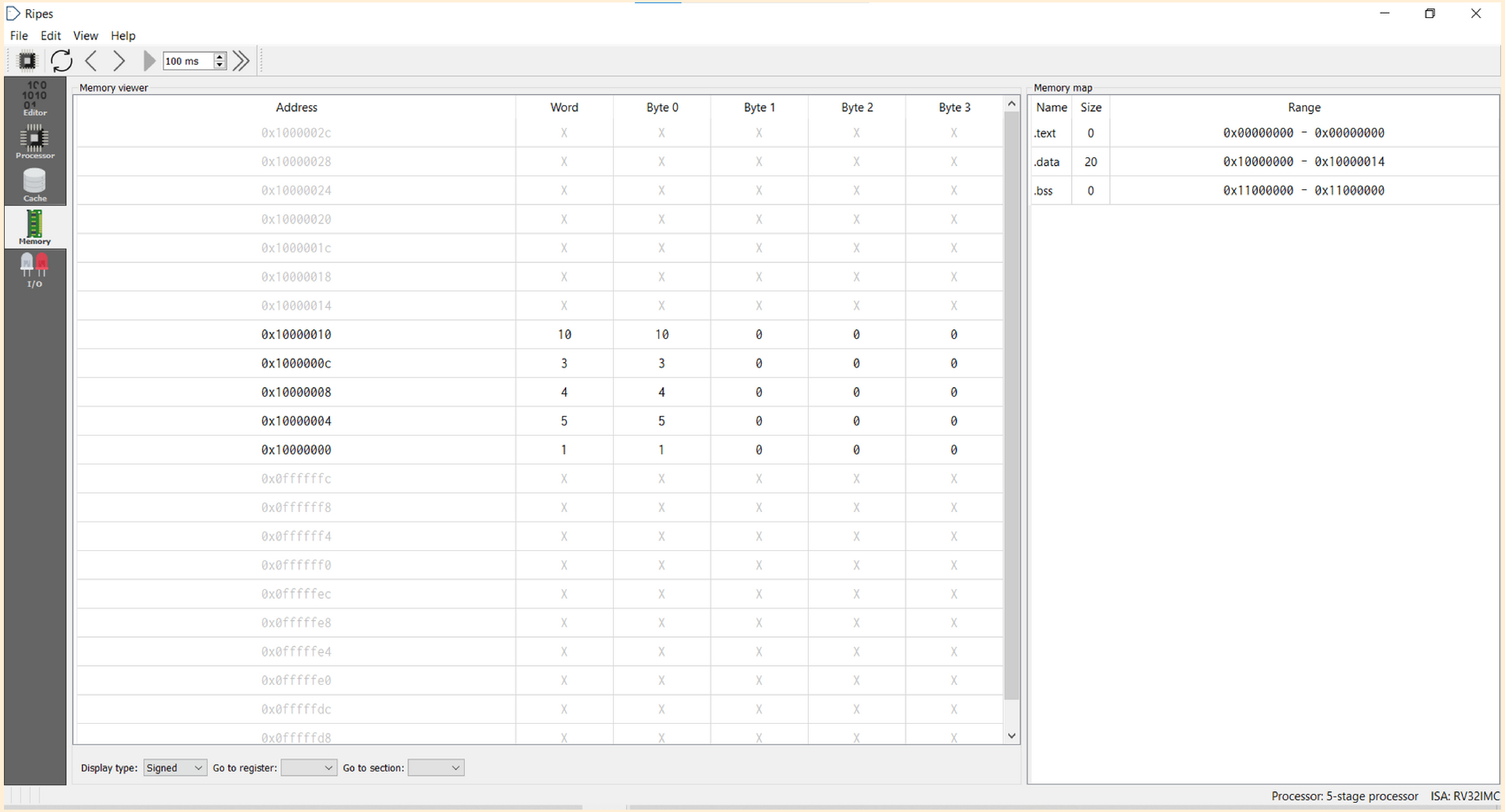
- j increments by 1
- x19 register holds the value of the length of the array.
- checks the condition if j is less than the value in x19 and then increments j
- finds the address of arr[i] and arr[min_id] and swapping function is performed.
- if arr[min_id] is less than arr[i]
- then i is incremented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

FLOWCHART



Loop
Loop
Start
Read the length of array in n, and the array in arr
Set i = 0
Is i < n-1
yes
No
Set min_index = i
Set j = i + 1
Is j < n
yes
No
Is arr[j] < arr[min_index]
min_index = j
J++
swap(arr[min_index], arr[i])
i++
print the sorted array
End

RESULTS-BEFORE



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	3	3	0	0	0
0x10000008	4	4	0	0	0
0x10000004	5	5	0	0	0
0x10000000	1	1	0	0	0
0x0ffffffc	X	X	X	X	X
0x0ffffff8	X	X	X	X	X
0x0ffffff4	X	X	X	X	X
0x0ffffff0	X	X	X	X	X
0x0fffffec	X	X	X	X	X
0x0fffffe8	X	X	X	X	X
0x0fffffe4	X	X	X	X	X
0x0fffffe0	X	X	X	X	X
0x0fffffdc	X	X	X	X	X
0x0fffffd8	X	X	X	X	X

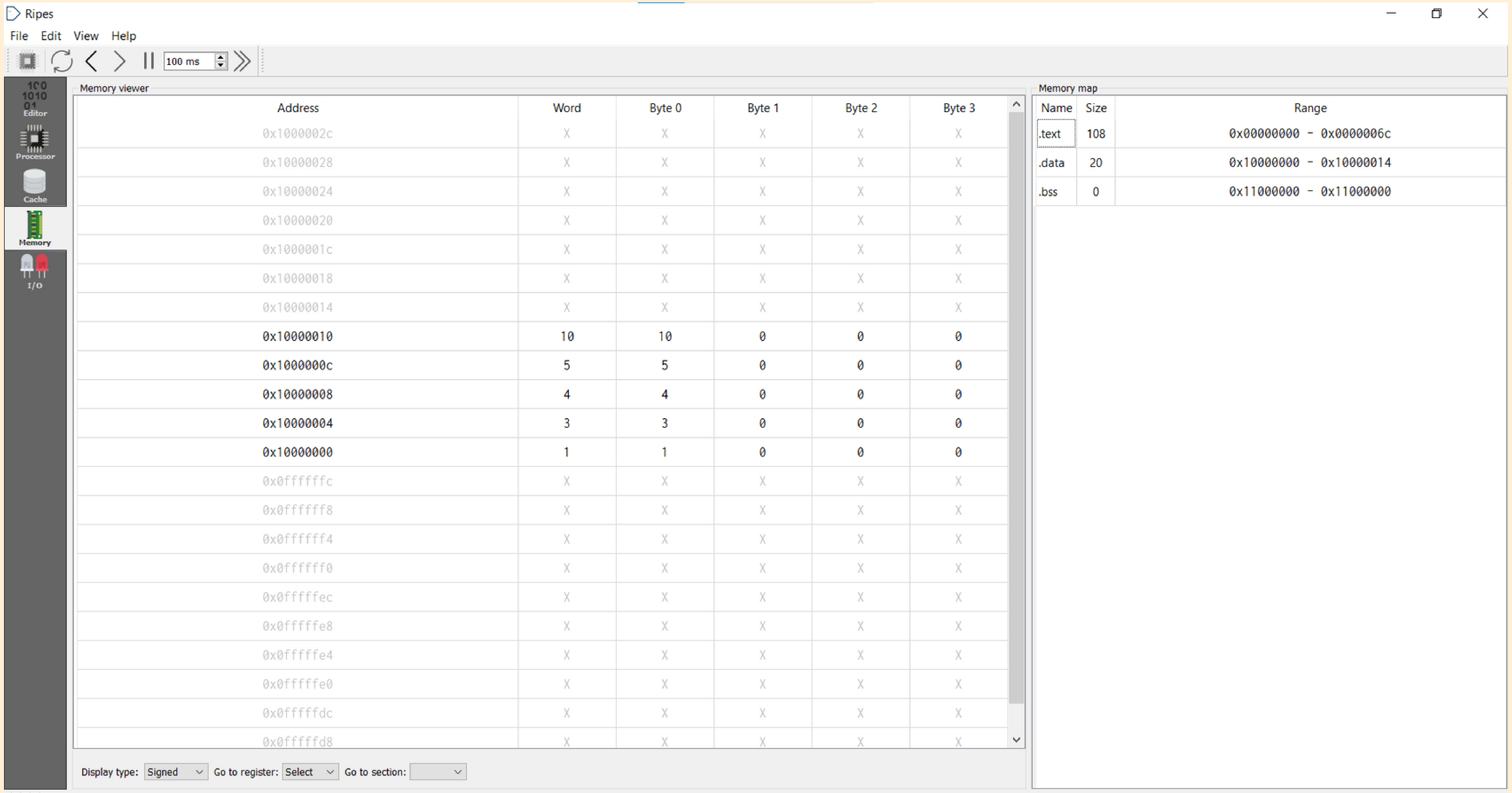
Display type: Signed   Go to register: Go to section:

Memory map

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

Processor: 5-stage processor   ISA: RV32IMC

RESULTS-AFTER



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
0x0ffffffc	X	X	X	X	X
0x0ffffff8	X	X	X	X	X
0x0ffffff4	X	X	X	X	X
0x0ffffff0	X	X	X	X	X
0x0fffffec	X	X	X	X	X
0x0fffffe8	X	X	X	X	X
0x0fffffe4	X	X	X	X	X
0x0fffffe0	X	X	X	X	X
0x0fffffdc	X	X	X	X	X
0x0fffffd8	X	X	X	X	X

Display type: Signed Go to register: Select Go to section:

Memory map

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

THANK YOU