```
#program to do binary search
# a0 = int arr[]
# a1 = int key
# a2 = int size
# t0 = mid
# t1 = left
# t2 = right

#Store [1,3,7,9,11,13,15,17,19,21,23] into the Data Memory

addi a0,x0,1
sw a0,0(x0)
addi a0,x0,3
sw a0,4(x0)
addi a0,x0,7
sw a0,8(x0)
addi a0,x0,9
sw a0,12(x0)
addi a0,x0,11
sw a0,16(x0)
addi a0,x0,13
sw a0,20(x0)
addi a0,x0,15
sw a0,24(x0)
addi a0,x0,17
sw a0,28(x0)
addi a0,x0,19
sw a0,32(x0)
addi a0,x0,21
```

```
sw a0,36(x0)

addi a0,x0,23

sw a0,40(x0)


#Set a0 register as address of first element of arr[]

addi a0,x0,0


#value of key : element to be searched

addi a1,x0,17


#Size of the arr[]

addi a2,x0,10


#Start the Binary Search

   addi    t1, zero, 0   # left = 0

   addi    t2, a2, -1   # right = size - 1

LOOP1:

# while loop

   blt     t2, t1, HALT     # right < left , break

   add     t0, t1, t2   # mid = left + right

   srai    t0, t0, 1       # mid = (left + right) / 2


   # Get the element at the midpoint

   slli    t4, t0, 2     # Scale the midpoint by 4

   add     t4, a0, t4   # Get the memory address of arr[mid]

   lw      t4, 0(t4)     # Dereference arr[mid]


   # See if the needle (a1) > arr[mid] (t3)

   bge     t4, a1, LOOP2    # if key <= t4, we need to check the next condition
```

```
    # If we get here, then the key is > arr[mid]
    addi    t1, t0, 1     # left = mid + 1
    jal     zero, LOOP1
LOOP2:
    beq     a1, t4, FOUND    # skip if key === arr[mid]
    # If we get here, then key < arr[mid]
    addi    t2, t0, -1   # right = mid - 1
    jal     zero, LOOP1
FOUND:
    # If we get here, then key == arr[mid]
    slli    sp, t0, 2     # Scale the midpoint by 4
    lw ra, 0(sp)
HALT:
    ecall
```