

COMPUTER ARCHITECTURE ASSIGNMENT – 2

We have implemented the following programs:

1. Finding the factorial of a number
2. Sorting an array using selection sort
3. Searching a number in an array using Binary Search

FACTORIAL OF A NUMBER

- **INTRODUCTION:**

Factorial of a number is defined as the product of all positive integers less than or equal to a given positive integer (Factorial of 0 is defined as 1).

Our program calculates the factorial of the given number. Assumption:

The input is a non-negative number.

- **PROGRAM EXPLANATION:**

In our program we have four labels : main, check, loop and exit.

main: Stores the necessary values of the input, memory address of input, the value of the iterator \$s1 and stores the value 1 in \$s2 which will be used to compute factorial. **check:**

Check if the iterator is less than our number or not and branch accordingly.

loop: Multiplies the current product value(\$s2) to the iterator \$s1 and then increments the iterator.

exit: Stores the result of factorial in data memory and exits from the program.

Instructions Used:

- Load Word (lw) ○ Store Word (sw) ○ Add immediate (addi) ○ Multiply (mul) ○ Branch on not equal (bne) ○ Jump (j)

• ASSEMBLY CODE:

```

1  main:
2
3      addi $t0, $0, 10          #t0 = n = 10
4      addi $t1, $0, 268500992  #t1 = memory address where n will be stored
5      sw $t0, 0($t1)           #M[t1] = t0 = 10
6      lw $s0, 0($t1)           #s0 = M[t1] = n = 10
7      addi $s1, $0, 1          #s1 = i = 1
8      addi $s2, $0, 1          #s2 = fact(n) = 1
9      j check
10
11  check:      #checks the loop condition
12
13      bne $s1, $s0, loop       #if (i!=n) jump to loop
14      mul $s2, $s2, $s0        #else fact = fact * n because the loop function will execute only till (n-1)
15      j exit
16
17  loop:
18      mul $s2, $s2, $s1        # fact = fact * i
19      addi $s1, $s1, 1         # i = i+1
20      j check
21
22  exit:
23      addi $t0, $0, 268501024  #t0 = address where the result will be stored
24      sw $s2, 0($t0)           #M[t0] = s2
25      addi $v0, $0, 10         # return 0
26      syscall

```

• TEXT SEGMENT:

| Text Segment | | | | |
|--------------------------|--------------------|---------|------------|---|
| Bkpt | Basic | Address | Code | Source |
| <input type="checkbox"/> | addi \$8,\$0,10 | 4194304 | 0x2008000a | 3: addi \$t0, \$0, 10 #t0 = n = 10 |
| <input type="checkbox"/> | lui \$1,4097 | 4194308 | 0x3c011001 | 4: addi \$t1, \$0, 268500992 #t1 = memory address where n will be st... |
| <input type="checkbox"/> | ori \$1,\$1,0 | 4194312 | 0x34210000 | |
| <input type="checkbox"/> | add \$9,\$0,\$1 | 4194316 | 0x00014820 | |
| <input type="checkbox"/> | sw \$8,0(\$9) | 4194320 | 0xad280000 | 5: sw \$t0, 0(\$t1) #M[t1] = t0 = 10 |
| <input type="checkbox"/> | lw \$16,0(\$9) | 4194324 | 0x8d300000 | 6: lw \$s0, 0(\$t1) #s0 = M[t1] = n = 10 |
| <input type="checkbox"/> | addi \$17,\$0,1 | 4194328 | 0x20110001 | 7: addi \$s1, \$0, 1 #s1 = i = 1 |
| <input type="checkbox"/> | addi \$18,\$0,1 | 4194332 | 0x20120001 | 8: addi \$s2, \$0, 1 #s2 = fact(n) = 1 |
| <input type="checkbox"/> | j 4194340 | 4194336 | 0x08100009 | 9: j check |
| <input type="checkbox"/> | bne \$17,\$16,2 | 4194340 | 0x16300002 | 13: bne \$s1, \$s0, loop #if (i!=n) jump to loop |
| <input type="checkbox"/> | mul \$18,\$18,\$16 | 4194344 | 0x72509002 | 14: mul \$s2, \$s2, \$s0 #else fact = fact * n because the loop function... |
| <input type="checkbox"/> | j 4194364 | 4194348 | 0x0810000f | 15: j exit |
| <input type="checkbox"/> | mul \$18,\$18,\$17 | 4194352 | 0x72519002 | 18: mul \$s2, \$s2, \$s1 # fact = fact * i |
| <input type="checkbox"/> | addi \$17,\$17,1 | 4194356 | 0x22310001 | 19: addi \$s1, \$s1, 1 # i = i+1 |
| <input type="checkbox"/> | j 4194340 | 4194360 | 0x08100009 | 20: j check |
| <input type="checkbox"/> | lui \$1,4097 | 4194364 | 0x3c011001 | 23: addi \$t0, \$0, 268501024 #t0 = address where the result will be stored |
| <input type="checkbox"/> | ori \$1,\$1,32 | 4194368 | 0x34210020 | |
| <input type="checkbox"/> | add \$8,\$0,\$1 | 4194372 | 0x00014020 | |
| <input type="checkbox"/> | sw \$18,0(\$8) | 4194376 | 0xad120000 | 24: sw \$s2, 0(\$t0) #M[t0] = s2 |
| <input type="checkbox"/> | addi \$2,\$0,10 | 4194380 | 0x2002000a | 25: addi \$v0, \$0, 10 # return 0 |
| <input type="checkbox"/> | syscall | 4194384 | 0x0000000c | 26: syscall |

• DATA SEGMENT:

| Data Segment | | | | | | | | |
|--------------|------------|------------|------------|-------------|-------------|-------------|-------------|-------------|
| Address | Value (+0) | Value (+4) | Value (+8) | Value (+12) | Value (+16) | Value (+20) | Value (+24) | Value (+28) |
| 268500992 | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 268501024 | 3628800 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 268501056 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 268501088 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 268501120 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 268501152 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 268501184 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 268501216 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 268501248 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 268501280 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 268501312 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 268501344 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 268501376 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 268501408 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 268501440 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 268501472 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

- MEMORY:**

| Registers | | | Coproc 1 | Coproc 0 |
|-----------|--------|------------|----------|----------|
| Name | Number | Value | | |
| \$zero | 0 | 0 | | |
| \$at | 1 | 268501024 | | |
| \$v0 | 2 | 10 | | |
| \$v1 | 3 | 0 | | |
| \$a0 | 4 | 0 | | |
| \$a1 | 5 | 0 | | |
| \$a2 | 6 | 0 | | |
| \$a3 | 7 | 0 | | |
| \$t0 | 8 | 268501024 | | |
| \$t1 | 9 | 268500992 | | |
| \$t2 | 10 | 0 | | |
| \$t3 | 11 | 0 | | |
| \$t4 | 12 | 0 | | |
| \$t5 | 13 | 0 | | |
| \$t6 | 14 | 0 | | |
| \$t7 | 15 | 0 | | |
| \$s0 | 16 | 10 | | |
| \$s1 | 17 | 10 | | |
| \$s2 | 18 | 3628800 | | |
| \$s3 | 19 | 0 | | |
| \$s4 | 20 | 0 | | |
| \$s5 | 21 | 0 | | |
| \$s6 | 22 | 0 | | |
| \$s7 | 23 | 0 | | |
| \$t8 | 24 | 0 | | |
| \$t9 | 25 | 0 | | |
| \$k0 | 26 | 0 | | |
| \$k1 | 27 | 0 | | |
| \$gp | 28 | 268468224 | | |
| \$sp | 29 | 2147479548 | | |
| \$fp | 30 | 0 | | |
| \$ra | 31 | 0 | | |
| pc | | 4194388 | | |

SELECTION SORT

- **INTRODUCTION:**

Selection sort is a sorting algorithm that works by repeatedly selecting the smallest (or largest) element from the unsorted portion of the array and moving it to the sorted portion of the array.

Our program sorts the array using the selection sort method.

- **PROGRAM EXPLANATION:**

Our program has 5 parts : main, outer_loop, cont_out, inner_loop, cont_in and exit.

main : Stores the necessary values, the integer 4 for alignment, the size of the array, the base address of the array and the iterator for the outer_loop which is \$s0.

outer_loop : Checks if you have iterated till the end of the loop. If yes, then branches to exit, else it stores the index of the minimum value obtained through inner_loop in \$s7. It also increments the iterator for the inner_loop which is \$s2.

cont_out : This swaps the minimum element to the desired place in the array (which is the outer loop iterator).

inner_loop: It calculates the next minimum element and then goes to cont_out which then swaps it. **cont_in**: Increments the inner iterator which is \$s2 and proceeds to inner_loop.

exit: Exits from the program.

Instructions Used:

- Load Word (lw) ○ Store Word (sw) ○ Add immediate (addi) ○ Multiply (mul) ○ Add (add)
- Branch on greater than equal (bge) ○ Jump (j)

- **ASSEMBLY CODE:**

```

1  .data
2      arr: .word 23,31,-9,180,99
3  .text
4
5  addi $s5,$zero,4           #store value of 4 to be used later for alignment
6  addi $t0,$0,5              #store size of array in $t0
7  la $t1,arr                 #store base address of array
8
9  #selection sorting code
10      add $s0,$zero,$zero    #counter for outer loop (i=0)
11      addi $s1,$t0,-1        # store len-1 in $s1
12  outer_loop: bge $s0,$s1, exit #check if i>=len-1 then break
13      add $s7,$s0,$zero      #min_pos=i
14      addi $s2,$s0,1         #counter for inner loop , j=i+1
15      j inner_loop
16
17      #swap min-pos to place
18  cont_out : mul $t3,$s5,$s0  #i*4 for alignment
19      add $t3,$t3,$t1        #store value of index i
20      lw $t4,0($t3)          #tmp= arr[i]
21
22      mul $t6,$s5,$s7         #min_pos*4 for alignment
23      add $t6,$t6,$t1        #store min_pos address val
24      lw $t5,0($t6)          #tmp2= arr[min_pos]
25
26      sw $t5,0($t3)          #arr[i] = tmp2 = arr[min_pos]
27      sw $t4,0($t6)          #arr[min_pos] = tmp = arr[i]
28
29      addi $s0,$s0,1         #i++
30      j outer_loop
31
32  inner_loop: bge $s2,$t0,cont_out #j>=n , break
33
34      mul $t3,$s5,$s2         #j*4 for alignment
35      add $t3,$t3,$t1        #store index j address
36      lw $t4,0($t3)          #tmp= arr[j]
37
38      mul $t6,$s5,$s7         #min_pos*4 for alignment
39      add $t6,$t6,$t1        #store min_pos address val
40      lw $t5,0($t6)          #tmp2= arr[min_pos]
41
42      bge $t4,$t5,cont_in     #if(arr[j]>=arr[min_pos], no change
43      add $s7,$s2,$zero      #min_pos=j;
44
45  cont_in: addi $s2,$s2,1     #j++
46      j inner_loop
47
48  exit: addi $v0,$0,10       #exit program
49      syscall
50

```


- **MEMORY:**

| Registers | Coproc 1 | Coproc 0 | |
|-----------|----------|------------|--|
| Name | Number | Value | |
| \$zero | 0 | 0 | |
| \$at | 1 | 0 | |
| \$v0 | 2 | 10 | |
| \$v1 | 3 | 0 | |
| \$a0 | 4 | 0 | |
| \$a1 | 5 | 0 | |
| \$a2 | 6 | 0 | |
| \$a3 | 7 | 0 | |
| \$t0 | 8 | 5 | |
| \$t1 | 9 | 268500992 | |
| \$t2 | 10 | 0 | |
| \$t3 | 11 | 268501004 | |
| \$t4 | 12 | 180 | |
| \$t5 | 13 | 99 | |
| \$t6 | 14 | 268501008 | |
| \$t7 | 15 | 0 | |
| \$s0 | 16 | 4 | |
| \$s1 | 17 | 4 | |
| \$s2 | 18 | 5 | |
| \$s3 | 19 | 0 | |
| \$s4 | 20 | 0 | |
| \$s5 | 21 | 4 | |
| \$s6 | 22 | 0 | |
| \$s7 | 23 | 4 | |
| \$t8 | 24 | 0 | |
| \$t9 | 25 | 0 | |
| \$k0 | 26 | 0 | |
| \$k1 | 27 | 0 | |
| \$gp | 28 | 268468224 | |
| \$sp | 29 | 2147479548 | |
| \$fp | 30 | 0 | |
| \$ra | 31 | 0 | |
| pc | | 4194448 | |

BINARY SEARCH

- **INTRODUCTION:**

Binary Search is a searching algorithm which is used in a sorted array by repeatedly dividing the search interval in half.

Our program searches for a particular number in an array of integers using binary search. If the number exists, the result is the corresponding index in the array; if not, the result is -1.

- **PROGRAM EXPLANATION:**

In this program, we have six parts: main, loop, else, found, not_found and exit.

main: Stores the necessary values, i.e., size of the array, the number to be searched (key), base address of array and the length of the array. We also declare lb and ub which is essential to perform the binary search. (NOTE: the boundaries of the array that we are working with at one point is defined by lower-bound and upper-bound).

loop: Checks if the lower-bound (lb) has exceeded upper-bound(ub) or not. If it has, our search is over. If not, it calculates the midpoint of the part of the array we are currently working with and stores the address of it in \$t4. If the element at index midpoint is equal to our key, we have found the value and we branch to found. Else if the element is less than key, we update upperbound to be (midpoint-1). Otherwise, we branch to the else label.

else: It changes the lower bound to (mid + 1) and goes back to loop for the next iteration.

found: Stores the index at which the element is found.

not_found: If the number is not there in the array, it stores the result as -1. **exit:**

Exits from the program.

Instructions Used:

- Load Word (lw) ○ Store Word (sw) ○ Load address (la) ○ Add immediate (addi) ○ Add (add)
- Shift right logical (srl)
- Shift left logical (sll) ○ Branch on equal (beq) ○ Branch on less than (blt) ○ Branch on greater than (bgt) ○ Jump (j)

- **ASSEMBLY CODE:**

```

1  #binary search
2  .data
3      arr: .word 11, 20, 34, 45, 56  #sorted array
4  .text
5
6  addi $s0,$zero,5          #store size of array in $s0
7  addi $s1,$zero,45         #number to be found
8  la $t1,arr                #store base address of array
9
10 addi $t7,$zero,0          #lowerbound=0
11 addi $t6,$s0,-1           #upperbound=size-1
12
13 loop:  bgt $t7,$t6,not_found #if lowerbound>upperbound, search is over
14        add $t5,$t7,$t6      #mid = lb+ub
15        srl $t5,$t5,1        #mid = lb+ub/2
16        sll $t4,$t5,2        #mid*4 for alignment
17        add $t4,$t4,$t1      #store address of arr[mid]
18        lw $t3,0($t4)        #$t3 = arr[mid]
19        beq $t3,$s1,found     #arr[mid] = key
20        blt $t3,$s1,else      #if arr[mid]<key
21        addi $t6,$t5,-1      #ub= mid-1
22        j loop
23
24 else:   addi $t7,$t5,1        #lb=mid+1
25        j loop
26
27 found:  addi $t0,$0,268501024 #t0 = address where the result will be stored
28        sw $t5,0($t0)         #M[t0] = t5 = index of element found
29        j exit
30
31 not_found:
32        addi $t5,$zero,-1      #to store -1 if no. not found
33        addi $t0,$0,268501024 #t0 = address where the result will be stored
34        sw $t5,0($t0)         #M[t0] = t5
35
36 exit:   addi $v0,$0,10        #exit program
37        syscall
38

```

- **TEXT SEGMENT:**

| Text Segment | | | | |
|--------------|---------|------------|--------------------|--|
| Bkpt | Address | Code | Basic | Source |
| | 4194364 | 0x20100005 | addi \$16,\$0,5 | 6: addi \$s0,\$zero,5 #store size of array in \$s0 |
| | 4194366 | 0x2011002d | addi \$17,\$0,45 | 7: addi \$s1,\$zero,45 #number to be found |
| | 4194372 | 0x3c011001 | lui \$1,4097 | 8: la \$t1,arr #store base address of array |
| | 4194316 | 0x34290000 | ori \$9,\$1,0 | |
| | 4194320 | 0x200f0000 | addi \$15,\$0,0 | 10: addi \$t7,\$zero,0 #lowerbound=0 |
| | 4194324 | 0x220effff | addi \$14,\$16,-1 | 11: addi \$t6,\$s0,-1 #upperbound=size-1 |
| | 4194328 | 0x01cf082a | slt \$1,\$14,\$15 | 13: loop: bgt \$t7,\$t6,not_found #if lowerbound>upperbound, search is over |
| | 4194332 | 0x14200011 | bne \$1,\$0,17 | |
| | 4194336 | 0x01ee6820 | add \$13,\$15,\$14 | 14: add \$t5,\$t7,\$t6 #mid = lb+ub |
| | 4194340 | 0x000d6842 | srl \$13,\$13,1 | 15: srl \$t5,\$t5,1 #mid = lb+ub/2 |
| | 4194344 | 0x000d6800 | sll \$12,\$13,2 | 16: sll \$t4,\$t5,2 #mid*4 for alignment |
| | 4194348 | 0x01996020 | add \$12,\$12,\$9 | 17: add \$t4,\$t4,\$t1 #store address of arr[mid] |
| | 4194352 | 0x8d8b0000 | lw \$11,0(\$t4) | 18: lw \$t3, 0(\$t4) #t3 = arr[mid] |
| | 4194356 | 0x11710006 | beq \$11,\$17,6 | 19: beq \$t3,\$s1,found #arr[mid] = key |
| | 4194360 | 0x0171082a | slt \$1,\$11,\$17 | 20: blt \$t3,\$s1,else #if arr[mid]<key |
| | 4194364 | 0x14200002 | bne \$1,\$0,2 | |
| | 4194368 | 0x21aeffff | addi \$14,\$13,-1 | 21: addi \$t6,\$t5,-1 #ub= mid-1 |
| | 4194372 | 0x06100006 | j 4194328 | 22: j loop |
| | 4194376 | 0x21af0001 | addi \$15,\$13,1 | 24: else: addi \$t7,\$t5,1 #lb=mid+1 |
| | 4194380 | 0x06100006 | j 4194328 | 25: j loop |
| | 4194384 | 0x3c011001 | lui \$1,4097 | 27: found: addi \$t0,\$0,268501024 #t0 = address where the result will be stored |
| | 4194386 | 0x34210020 | ori \$1,\$1,32 | |
| | 4194392 | 0x00014020 | add \$9,\$0,\$1 | |
| | 4194396 | 0xad0d0000 | sw \$13,0(\$9) | 28: sw \$t5,0(\$t0) #M[t0] = t5 = index of element found |
| | 4194400 | 0x0610001e | j 4194424 | 29: j exit |
| | 4194404 | 0x200dffff | addi \$13,\$0,-1 | 32: addi \$t5,\$zero,-1 #t0 store -1 if no. not found |
| | 4194408 | 0x3c011001 | lui \$1,4097 | 33: addi \$t0,\$0,268501024 #t0 = address where the result will be stored |
| | 4194412 | 0x34210020 | ori \$1,\$1,32 | |
| | 4194416 | 0x00014020 | add \$9,\$0,\$1 | |
| | 4194420 | 0xad0d0000 | sw \$13,0(\$9) | 34: sw \$t5,0(\$t0) #M[t0] = t5 |
| | 4194424 | 0x20020008 | addi \$2,\$0,10 | 37: addi \$v0,\$0,10 #exit program |
| | 4194428 | 0x0000000c | syscall | 38: syscall |

• DATA SEGMENT:

| Data Segment | | | | | | | | | | |
|--------------|------------|------------|------------|-------------|-------------|-------------|-------------|-------------|--|--|
| Address | Value (+0) | Value (+4) | Value (+8) | Value (+12) | Value (+16) | Value (+20) | Value (+24) | Value (+28) | | |
| 268500992 | 11 | 20 | 34 | 45 | 56 | 0 | 0 | 0 | | |
| 268501024 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | |
| 268501056 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | |
| 268501088 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | |
| 268501120 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | |
| 268501152 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | |
| 268501184 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | |
| 268501216 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | |
| 268501248 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | |
| 268501280 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | |
| 268501312 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | |
| 268501344 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | |
| 268501376 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | |
| 268501408 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | |
| 268501440 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | |
| 268501472 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | |

• MEMORY:

| Registers | Coproc 1 | Coproc 0 |
|-----------|----------|------------|
| Name | Number | Value |
| \$zero | 0 | 0 |
| \$at | 1 | 268501024 |
| \$v0 | 2 | 10 |
| \$v1 | 3 | 0 |
| \$a0 | 4 | 0 |
| \$a1 | 5 | 0 |
| \$a2 | 6 | 0 |
| \$a3 | 7 | 0 |
| \$t0 | 8 | 268501024 |
| \$t1 | 9 | 268500992 |
| \$t2 | 10 | 0 |
| \$t3 | 11 | 45 |
| \$t4 | 12 | 268501004 |
| \$t5 | 13 | 3 |
| \$t6 | 14 | 4 |
| \$t7 | 15 | 3 |
| \$s0 | 16 | 5 |
| \$s1 | 17 | 45 |
| \$s2 | 18 | 0 |
| \$s3 | 19 | 0 |
| \$s4 | 20 | 0 |
| \$s5 | 21 | 0 |
| \$s6 | 22 | 0 |
| \$s7 | 23 | 0 |
| \$t8 | 24 | 0 |
| \$t9 | 25 | 0 |
| \$k0 | 26 | 0 |
| \$k1 | 27 | 0 |
| \$gp | 28 | 268468224 |
| \$sp | 29 | 2147479548 |
| \$fp | 30 | 0 |
| \$ra | 31 | 0 |
| pc | | 4194432 |

ASSUMPTIONS IN PROCESSOR:

- 1) We have defined the ALUOp of the bne instruction as 11.
- 2) We have defined the ALU control input code of srl as '101' and sll as '110'.

RESULT (FROM PROCESSOR)

- FACTORIAL

[illegible]

```
Register file:
[0: 0, 1: 268500992, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 10, 9: 268500992, 10: 0, 11: 0, 12: 0, 13: 0, 14: 0, 15: 0, 16: 10, 17: 6, 18: 120, 19: 0, 20: 0, 21: 0, 22: 0, 23: 0, 24: 0, 25: 0, 26: 0, 27: 0, 28: 268648224, 29: 2147479548, 30: 0, 31: 0]

Data Memory:
[268500992: 10, 268501024: 0]
```

[illegible]

- BINARY SEARCH

```

PC: 4194304
Register file:
{0: 0, 1: 0, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 9: 0, 10: 0, 11: 0, 12: 0, 13: 0, 14: 0, 15: 0, 16: 5, 17: 0, 18: 0, 19: 0, 20: 0, 21: 0, 22: 0, 23: 0, 24: 0, 25: 0, 26: 0, 27: 0, 28: 268468224, 29: 2147479548, 30: 0, 31: 0}
Data Memory:
{268500992: 11, 268500996: 20, 268501000: 34, 268501004: 45, 268501008: 56, 268501024: 0}

PC: 4194308
Register file:
{0: 0, 1: 0, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 9: 0, 10: 0, 11: 0, 12: 0, 13: 0, 14: 0, 15: 0, 16: 5, 17: 45, 18: 0, 19: 0, 20: 0, 21: 0, 22: 0, 23: 0, 24: 0, 25: 0, 26: 0, 27: 0, 28: 268468224, 29: 2147479548, 30: 0, 31: 0}
Data Memory:
{268500992: 11, 268500996: 20, 268501000: 34, 268501004: 45, 268501008: 56, 268501024: 0}

PC: 4194312
Register file:
{0: 0, 1: 268500992, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 9: 0, 10: 0, 11: 0, 12: 0, 13: 0, 14: 0, 15: 0, 16: 5, 17: 45, 18: 0, 19: 0, 20: 0, 21: 0, 22: 0, 23: 0, 24: 0, 25: 0, 26: 0, 27: 0, 28: 268468224, 29: 2147479548, 30: 0, 31: 0}
Data Memory:
{268500992: 11, 268500996: 20, 268501000: 34, 268501004: 45, 268501008: 56, 268501024: 0}

PC: 4194316
Register file:
{0: 0, 1: 268500992, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 9: 268500992, 10: 0, 11: 0, 12: 0, 13: 0, 14: 0, 15: 0, 16: 5, 17: 45, 18: 0, 19: 0, 20: 0, 21: 0, 22: 0, 23: 0, 24: 0, 25: 0, 26: 0, 27: 0, 28: 268468224, 29: 2147479548, 30: 0, 31: 0}
Data Memory:
{268500992: 11, 268500996: 20, 268501000: 34, 268501004: 45, 268501008: 56, 268501024: 0}

PC: 4194320
Register file:
{0: 0, 1: 268500992, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 9: 268500992, 10: 0, 11: 0, 12: 0, 13: 0, 14: 0, 15: 0, 16: 5, 17: 45, 18: 0, 19: 0, 20: 0, 21: 0, 22: 0, 23: 0, 24: 0, 25: 0, 26: 0, 27: 0, 28: 268468224, 29: 2147479548, 30: 0, 31: 0}
Data Memory:
{268500992: 11, 268500996: 20, 268501000: 34, 268501004: 45, 268501008: 56, 268501024: 0}

PC: 4194324
Register file:
{0: 0, 1: 268500992, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 9: 268500992, 10: 0, 11: 0, 12: 0, 13: 0, 14: 0, 15: 0, 16: 5, 17: 45, 18: 0, 19: 0, 20: 0, 21: 0, 22: 0, 23: 0, 24: 0, 25: 0, 26: 0, 27: 0, 28: 268468224, 29: 2147479548, 30: 0, 31: 0}
Data Memory:
{268500992: 11, 268500996: 20, 268501000: 34, 268501004: 45, 268501008: 56, 268501024: 0}

```

```
PC: 4190328  
Register file:  
[0: 0, 1: 0, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 9: 268500992, 10: 0, 11: 0, 12: 0, 13: 0, 14: 4, 15: 0, 16: 5, 17: 45, 18: 0, 19: 0, 20: 0, 21: 0, 22: 0, 23: 0, 24: 0, 25: 0, 26: 0, 27: 0, 28: 26860224, 29: 2147479548, 30: 0, 31: 0]  
Data Memory:  
[268500992: 11, 268500996: 20, 268501000: 34, 268501004: 45, 268501008: 56, 268501024: 0]  
  
PC: 4190332  
Register file:  
[0: 0, 1: 0, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 9: 268500992, 10: 0, 11: 0, 12: 0, 13: 0, 14: 4, 15: 0, 16: 5, 17: 45, 18: 0, 19: 0, 20: 0, 21: 0, 22: 0, 23: 0, 24: 0, 25: 0, 26: 0, 27: 0, 28: 26860224, 29: 2147479548, 30: 0, 31: 0]  
Data Memory:  
[268500992: 11, 268500996: 20, 268501000: 24, 268501004: 45, 268501008: 56, 268501024: 0]  
  
PC: 4190336  
Register file:  
[0: 0, 1: 0, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 9: 268500992, 10: 0, 11: 0, 12: 0, 13: 0, 14: 4, 15: 0, 16: 5, 17: 45, 18: 0, 19: 0, 20: 0, 21: 0, 22: 0, 23: 0, 24: 0, 25: 0, 26: 0, 27: 0, 28: 26860224, 29: 2147479548, 30: 0, 31: 0]  
Data Memory:  
[268500992: 11, 268500996: 20, 268501000: 34, 268501004: 45, 268501008: 56, 268501024: 0]  
  
PC: 4190340  
Register file:  
[0: 0, 1: 0, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 9: 268500992, 10: 0, 11: 0, 12: 0, 13: 0, 14: 4, 15: 0, 16: 5, 17: 45, 18: 0, 19: 0, 20: 0, 21: 0, 22: 0, 23: 0, 24: 0, 25: 0, 26: 0, 27: 0, 28: 26860224, 29: 2147479548, 30: 0, 31: 0]  
Data Memory:  
[268500992: 11, 268500996: 20, 268501000: 34, 268501004: 45, 268501008: 56, 268501024: 0]  
  
PC: 4190344  
Register file:  
[0: 0, 1: 0, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 9: 268500992, 10: 0, 11: 0, 12: 0, 13: 0, 14: 4, 15: 0, 16: 5, 17: 45, 18: 0, 19: 0, 20: 0, 21: 0, 22: 0, 23: 0, 24: 0, 25: 0, 26: 0, 27: 0, 28: 26860224, 29: 2147479548, 30: 0, 31: 0]  
Data Memory:  
[268500992: 11, 268500996: 20, 268501000: 34, 268501004: 45, 268501008: 56, 268501024: 0]
```

R.
3

- SELECTION SORT

```
Data Memory:
{268500992: 23, 268500996: 31, 268501000: -9, 268501004: 180, 268501008: 99}
```



```
PC: 4194408
Data Memory:
```

```
PC: 4194424
Data Memory:
{268500992: -9, 268500996: 31, 268501000: 23, 268501004: 180, 268501008: 99}
```



```
Result:
{268500992: -9, 268500996: 23, 268501000: 31, 268501004: 99, 268501008: 180}
```