Input Code

```
void selectionSort(int arr[], int n) {
    int i, j, min_idx;

    for (i = 0; i < n - 1; i++) {
        min_idx = i;
        for (j = i + 1; j < n; j++) {
            if (arr[j] < arr[min_idx]) {
                min_idx = j;
            }
        }

        int temp = arr[min_idx];
        arr[min_idx] = arr[i];
        arr[i] = temp;
    }
}

int main() {
    int arr[] = {64, 25, 12, 22};
    int n = sizeof(arr) / sizeof(arr[0]);

    selectionSort(arr, n);

    return 0;
}
```

3-Address Code (Unoptimized)

```
selectionSort:
    i = 0
L1: if i >= n - 1 goto L6
    min_idx = i
    j = i + 1
L2: if j >= n goto L5
    t1 = j * sizeof(int)
    t2 = arr + t1
    t3 = *t2
    t4 = min_idx * sizeof(int)
    t5 = arr + t4
    t6 = *t5
    if t3 >= t6 goto L4
    min_idx = j
L4: j = j + 1
    goto L2
L5: t7 = min_idx * sizeof(int)
    t8 = arr + t7
    temp = *t8
    t9 = i * sizeof(int)
    t10 = arr + t9
    *t8 = *t10
    *t10 = temp
    i = i + 1
    goto L1
L6: return

main:
    n = 4
    t11 = &64
    arr = t11
    t12 = &25
    *(arr + sizeof(int)) = t12
    t13 = &12
    *(arr + 2 * sizeof(int)) = t13
    t14 = &22
    *(arr + 3 * sizeof(int)) = t14
    call selectionSort
    return 0
```

3-address code (Optimized)

```
selectionSort:
    i = 0
L1: if i >= n - 1 goto L6
    min_idx = i
    j = i + 1
    t1 = i * sizeof(int)
    t2 = arr + t1
    t6 = *t2
L2: if j >= n goto L5
    t3 = j * sizeof(int)
    t4 = arr + t3
    t5 = *t4
    if t5 >= t6 goto L4
    min_idx = j
    t6 = t5
L4: j = j + 1
    goto L2
L5: t7 = min_idx * sizeof(int)
    t8 = arr + t7
    temp = *t8
    *t8 = *t2
    *t2 = temp
    i = i + 1
    goto L1
L6: return

main:
    n = 4
    arr = &64
    *(arr + sizeof(int)) = &25
    *(arr + 2 * sizeof(int)) = &12
    *(arr + 3 * sizeof(int)) = &22
    call selectionSort
    return 0
```

Machine Code

```
selectionSort:
    pushq %rbp
    movq %rsp, %rbp
    movq %rdi, - 24(%rbp)        ; arr
    movl %esi, - 28(%rbp)        ; n
    movl $0, - 4(%rbp)           ; i = 0
    jmp .L1
.L1:
    movl - 4(%rbp), %eax
    cmpl - 28(%rbp), %eax        ; i cmp n - 1
    jge .L6
    movl - 4(%rbp), %eax         ; min_idx = i
    movl %eax, - 8(%rbp)
    movl - 4(%rbp), %eax
    addl $1, %eax                ; j = i + 1
    movl %eax, - 12(%rbp)
    movl - 4(%rbp), %eax
    cltq
    leaq 0(, %rax, 4), %rdx      ; t1 = i * sizeof(int)
    movq - 24(%rbp), %rax
    addq %rdx, %rax              ; t2 = arr + t1
    movl (%rax), %eax            ; t6 = * t2
    movl %eax, - 16(%rbp)
.L2:
    movl - 12(%rbp), %eax
    cmpl - 28(%rbp), %eax        ; j cmp n
    jge .L5
    movl - 12(%rbp), %eax
    cltq
    leaq 0(, %rax, 4), %rdx      ; t3 = j * sizeof(int)
    movq - 24(%rbp), %rax
    addq %rdx, %rax              ; t4 = arr + t3
    movl (%rax), %eax            ; t5 = * t4
    cmpl %eax, - 16(%rbp)        ; t5 cmp t6
    jle .L4
    movl - 12(%rbp), %eax        ; min_idx = j
    movl %eax, - 8(%rbp)
    movl (%rax), %eax            ; t6 = t5
    movl %eax, - 16(%rbp)
.L4:
    addl $1, - 12(%rbp)          ; j = j + 1
    jmp .L2
.L5:
    movl - 8(%rbp), %eax
    cltq
    leaq 0(, %rax, 4), %rdx      ; t7 = min_idx * sizeof(int)
    movq - 24(%rbp), %rax
    addq %rdx, %rax              ; t8 = arr + t7
    movl (%rax), %eax            ; temp = * t8
    movl %eax, - 20(%rbp)
    movl - 4(%rbp), %eax
    cltq
    leaq 0(, %rax, 4), %rdx      ; t1 = i * sizeof(int)
    movq - 24(%rbp), %rax
    addq %rdx, %rax              ; t2 = arr + t1
    movl - 16(%rbp), %edx
    movl %edx, (%rax)            ; * t2 = * t8
    movl - 8(%rbp), %eax
    cltq
    leaq 0(, %rax, 4), %rdx      ; t7 = min_idx * sizeof(int)
```

```
    movq -24(%rbp), %rax
    addq %rdx, %rax             ; t8 = arr + t7
    movl -20(%rbp), %edx
    movl %edx, (%rax)           ; * t8 = temp
    addl $1, -4(%rbp)           ; i = i + 1
    jmp .L1
.L6:
    nop
    popq %rbp
    ret

main:
    pushq %rbp
    movq %rsp, %rbp
    subq $32, %rsp
    movl $64, -32(%rbp)
    movl $25, -28(%rbp)
    movl $12, -24(%rbp)
    movl $22, -20(%rbp)
    movl $4, -4(%rbp)           ; n = 4
    leaq -32(%rbp), %rax        ; arr = &64
    movl -4(%rbp), %edx
    movl %edx, %esi             ; push n
    movq %rax, %rdi             ; push arr
    call selectionSort
    movl $0, %eax
    leave
    ret
```