

Consider the following C code

```
void quicksort(m, n)
int m, n;
{
    int I, j;
    if (n <= m ) return;
    /* fragment begins here */
    i = m-1; j = n; v = a[n];
    while(1) {
        do i = i+1; while( a[i] < v );
        do j = j-1; while( a[j] > v );
        if( i >= j ) break;
        x = a[i]; a[i] = a[j];    a[j] = x;
    }
    x = a[i]; a[i] = a[n]; a[n]= x;
    /* fragment ends here */
    quicksort(m, j); quicksort(i+1, n);
}
```

Augmented 3AC

An augmented 3 address code language to simplify the code...

Let a be an array of integers starting at byte address a_0

$a[\text{add}]$ on the left-hand-side of an assignment is the address $a_0 + \text{add}$

$a[\text{add}]$ on the right-hand-side of an assignment is the value of the element of the array at address $a_0 + \text{add}$

Since integers are stored in 4 bytes the offset address of an element $a[i]$ is $4 * i$

Three Address Code of Quick Sort

1	$i = m - 1$
2	$j = n$
3	$t_1 = 4 * n$
4	$v = a[t_1]$
5	$i = i + 1$
6	$t_2 = 4 * i$
7	$t_3 = a[t_2]$
8	if $t_3 < v$ goto (5)
9	$j = j - 1$
10	$t_4 = 4 * j$
11	$t_5 = a[t_4]$
12	if $t_5 > v$ goto (9)
13	if $i \geq j$ goto (23)
14	$t_6 = 4 * i$
15	$x = a[t_6]$

16	$t_7 = 4 * I$
17	$t_8 = 4 * j$
18	$t_9 = a[t_8]$
19	$a[t_7] = t_9$
20	$t_{10} = 4 * j$
21	$a[t_{10}] = x$
22	goto (5)
23	$t_{11} = 4 * I$
24	$x = a[t_{11}]$
25	$t_{12} = 4 * i$
26	$t_{13} = 4 * n$
27	$t_{14} = a[t_{13}]$
28	$a[t_{12}] = t_{14}$
29	$t_{15} = 4 * n$
30	$a[t_{15}] = x$

Exercise 1

1. Identify the basic blocks of the 3 address code of the previous slide
2. Build the control flow diagram
3. Apply as many optimizations you can to that code, pointing out which one you apply, and in which order
4. Insert your solution in the wiki page, and discuss it with the rest of the class