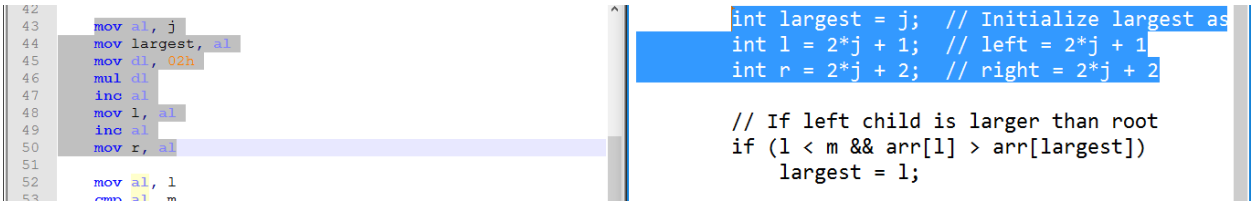


```
42
43      mov al, j
44      mov largest, al
45      mov dl, 02h
46      mul dl
47      inc al
48      mov l, al
49      inc al
50      mov r, al
51
52      mov al, l
```

```
int largest = j;  // Initialize largest as
int l = 2*j + 1;  // left = 2*j + 1
int r = 2*j + 2;  // right = 2*j + 2

// If left child is larger than root
if (l < m && arr[l] > arr[largest])
    largest = l;
```

این دو قسمت که هایلایت شده مربوط به همدیگر می باشد.

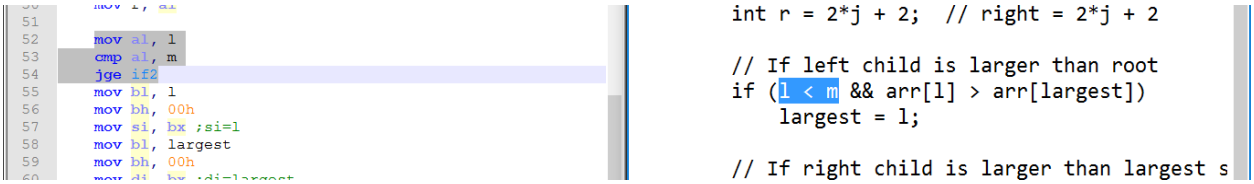
خط 43 و 44: چون مستقیما نمی توان از مموری ریخت تو مموری اول J رو میریزه تو ثبات al بعد از al میریزه تو largest

خط 45: بعد MUL نمیتوان از عدد استفاده کرد پس عدد 2h رو میریزیم توی dl بعد ضرب میکنیم

همیشه عملیات MUL، پاسخ ضرب را در AL میریزد (AL الان مقدار 2*j را داراست)

خط 47 مقدار AL را یک واحد زیاد می کند (AL الان مقدار 2*j+1 را داراست) و خط 48 توی L میریزه

خط 49 دوباره AL را یک واحد اضافه می کند (AL الان مقدار 2*j+2 را داراست) و در R میریزد



```
51
52      mov al, l
53      cmp al, m
54      jge if2
55      mov bl, l
56      mov bh, 00h
57      mov si, bx ;si=l
58      mov bl, largest
59      mov bh, 00h
```

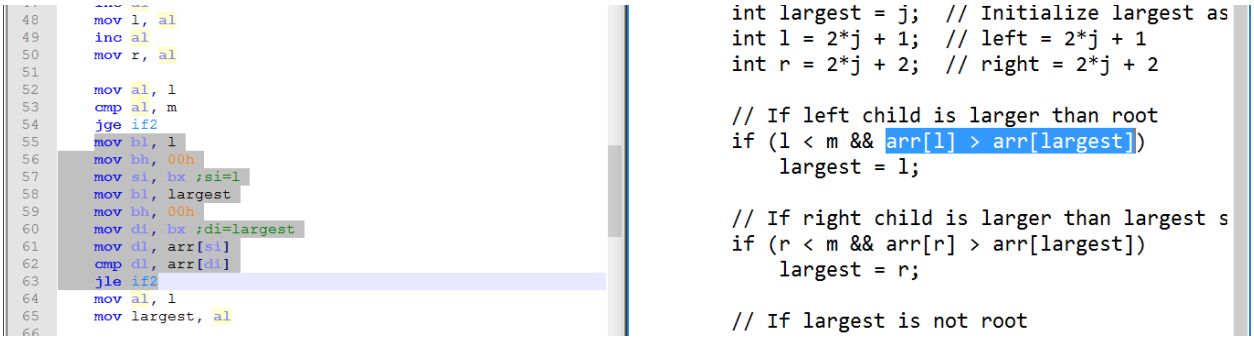
```
int r = 2*j + 2;  // right = 2*j + 2

// If left child is larger than root
if (l < m && arr[l] > arr[largest])
    largest = l;

// If right child is larger than largest s
```

این دو قسمت مربوط به هم هستند

l رو با m مقایسه میکنه اگه کوچکتر بود ادامه میده (میره خط بعد) ولی اگه بزرگتر یا مساوی بود وارد شرط نمیشه و میره شرط بعدی



```
48      mov l, al
49      inc al
50      mov r, al
51
52      mov al, l
53      cmp al, m
54      jge if2
55      mov bl, l
56      mov bh, 00h
57      mov si, bx ;si=l
58      mov bl, largest
59      mov bh, 00h
60      mov di, bx ;di=largest
61      mov dl, arr[si]
62      cmp dl, arr[di]
63      jle if2
64      mov al, l
65      mov largest, al
```

```
int largest = j;  // Initialize largest as
int l = 2*j + 1;  // left = 2*j + 1
int r = 2*j + 2;  // right = 2*j + 2

// If left child is larger than root
if (l < m && arr[l] > arr[largest])
    largest = l;

// If right child is larger than largest s
if (r < m && arr[r] > arr[largest])
    largest = r;

// If largest is not root
```

این دو قسمت مربوط به همند

توضیحات مانند عکس قبل

```
52      mov al, l
53      cmp al, m
54      jge if2
55      mov bl, l
56      mov bh, 00h
57      mov si, bx ;si=l
58      mov bl, largest
59      mov bh, 00h
60      mov di, bx ;di=largest
61      mov dl, arr[si]
62      cmp dl, arr[di]
63      jle if2
64      mov al, l
65      mov largest, al
66
67      if2:
68          mov al, r
```

```
    // If left child is larger than root
    if (l < m && arr[l] > arr[largest])
        largest = l;

    // If right child is larger than largest s
    if (r < m && arr[r] > arr[largest])
        largest = r;

    // If largest is not root
    if (largest != j)
    {
```

اگر دو شرط قبل همزمان برقرار بود این قسمت اجرا می شود

```
65      mov largest, al
66
67      if2:
68          mov al, r
69          cmp al, m
70          jge if3
71          mov bl, r
72          mov bh, 00h
73          mov si, bx ;si=r
74          mov bl, largest
75          mov bh, 00h
76          mov di, bx ;di=largest
77          mov dl, arr[si]
78          cmp dl, arr[di]
79          jle if3
80          mov al, r
81          mov largest, al
82
83      if3:
```

```
    int largest = j;  // Initialize largest as
    int l = 2*j + 1;  // left = 2*j + 1
    int r = 2*j + 2;  // right = 2*j + 2

    // If left child is larger than root
    if (l < m && arr[l] > arr[largest])
        largest = l;

    // If right child is larger than largest s
    if (r < m && arr[r] > arr[largest])
        largest = r;

    // If largest is not root
    if (largest != j)
    {
        int swap = arr[j];
```

این هم مثل شرط قبل ...

```
80          mov al, r
81          mov largest, al
82
83      if3:
84          mov bl, largest
85          cmp bl, j
86          je endif3
87
88          mov bl, j
89          mov bh, 00h
90          mov si, bx;j
91          mov dl, arr[si];arr[j]
92          mov bl, largest
93          mov bh, 00h
94          mov di, bx;largest
95          mov dh, arr[di]
96          mov arr[si], dh
97          mov arr[di], dl
98
99          mov al, largest
```

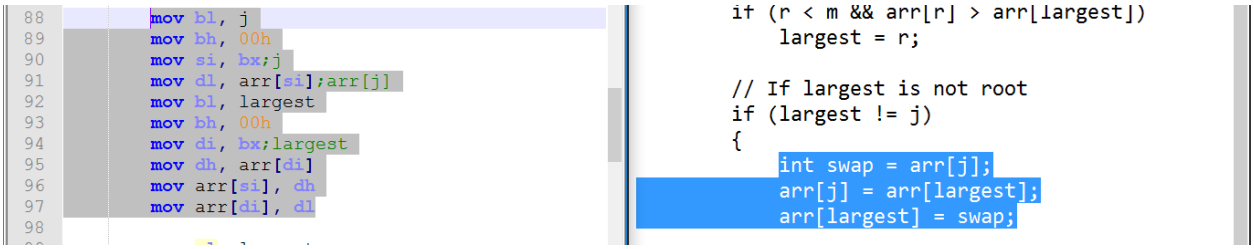
```
    // If left child is larger than root
    if (l < m && arr[l] > arr[largest])
        largest = l;

    // If right child is larger than largest s
    if (r < m && arr[r] > arr[largest])
        largest = r;

    // If largest is not root
    if (largest != j)
    {
        int swap = arr[j];
        arr[j] = arr[largest];
        arr[largest] = swap;
```

این دو قسمت مربوط به هم است

اگر largest و j برابر نباشند ادامه می دهد (خط 87 اجرا می شود) اگر برابر باشند به پایان شرط می رود (وارد شرط نمی شود.)



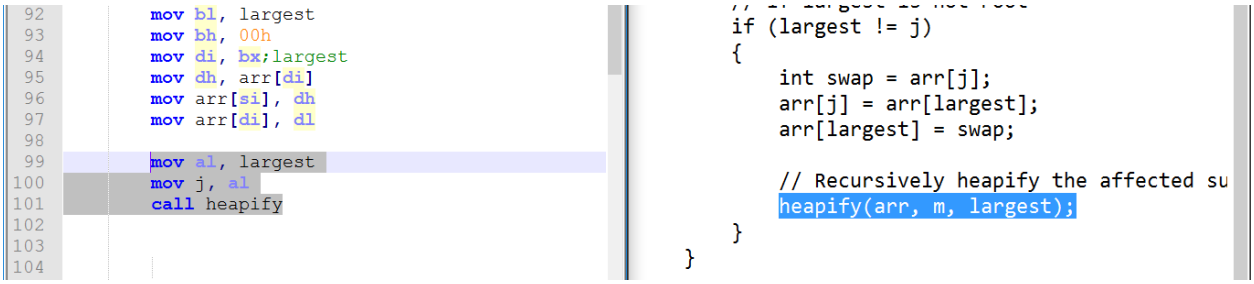
```
88          mov bl, j
89          mov bh, 00h
90          mov si, bx;j
91          mov dl, arr[si];arr[j]
92          mov bl, largest
93          mov bh, 00h
94          mov di, bx;largest
95          mov dh, arr[di]
96          mov arr[si], dh
97          mov arr[di], dl
98
```

```
        if (r < m && arr[r] > arr[largest])
            largest = r;

        // If largest is not root
        if (largest != j)
        {
            int swap = arr[j];
            arr[j] = arr[largest];
            arr[largest] = swap;
```

برای آدرس دهی حتما باید یا از bx یا si یا di استفاده کرد که ثبات 16 بیتی هستند ولی j و largest 8 بیتی هستند 8 بیتی هم نمیشه ریخت تو 16 بیتی

پس اول تو bl میریزیم بعد bx که 16 بیتیه میریزیم تو si



```
92          mov bl, largest
93          mov bh, 00h
94          mov di, bx;largest
95          mov dh, arr[di]
96          mov arr[si], dh
97          mov arr[di], dl
98
99          mov al, largest
100         mov j, al
101         call heapify
102
103
104
```

```
        if (largest != j)
        {
            int swap = arr[j];
            arr[j] = arr[largest];
            arr[largest] = swap;

            // Recursively heapify the affected su
            heapify(arr, m, largest);
        }
    }
```

مقدار m (ورودی متد) که تغییر نکرده همون بود

ولی ورودی دیگه متد که j بود این دفعه با Largest فراخوانی میشه

پس باید مقدار J رو برابر مقدار largest کرد بعد متد رو فراخوانی کرد

توضیح بیشتر اگه توضیح بالا رو متوجه نشدید:

متد سطح بالا: void heapify(int arr[], int m, int j)
پارامتر اولش که همون آرایمونه که ثابته همیشه

پارامتر دوم m که دفعه قبل هم با همین m فراخوانی کردیم پس مقدارش همونه

ولی هنگام فراخوانی پارامتر j مقدارش شده largest

(امیدوارم متوجه شده باشین، نشدین بگید)