

## Exam 3 - Problem 2



## Exam 3 - Problem 2



Given this code in C:

```
int heap (int& b, char c, int l) {  
  
    *b = *b + 1;  
    int f = l - 1;  
    int n = f + *b;  
  
    return n;  
}
```

Translate the subroutine `heap` to x86.



## Exam 3 - Problem 2

```
int heap (int& b, char c, int l) {
```

```
    *b = *b + 1;  
    int f = l - 1;  
    int n = f + *b;
```

```
    return n;
```

```
}
```

### # Function prologue

```
push %ebp
```

```
mov %esp, %ebp
```

### # Allocate space on the stack for local variables

```
sub $8, %esp      # 4 bytes for f, 4 bytes for n
```

### # Load the parameters onto the stack

```
mov 8(%ebp), %eax  # %eax = &b
```

```
mov (%eax), %eax   # %eax = *b
```

```
mov 12(%ebp), %cl  # Load the value of c
```

```
mov 16(%ebp), %edx # Load the value of l
```



Part 1/2



## Exam 3 - Problem 2

```
int heap (int& b, char c, int l) {
```

```
    *b = *b + 1;  
    int f = l - 1;  
    int n = f + *b;  
  
    return n;
```

```
}
```

**# Increment the value of b and store it back in b**

```
add $1, %eax  
mov %eax, (%ebx)
```

**# Calculate the value of n**

```
sub $1, %edx          # %edx = l - 1  
add %eax, %edx        # %edx = f + *b  
mov %edx, -4(%ebp)    # n = f + *b
```

**# Load n into %eax and return**

```
mov -4(%ebp), %eax
```

**# Function epilogue**

```
mov %ebp, %esp  
pop %ebp  
ret
```



Part 2/2