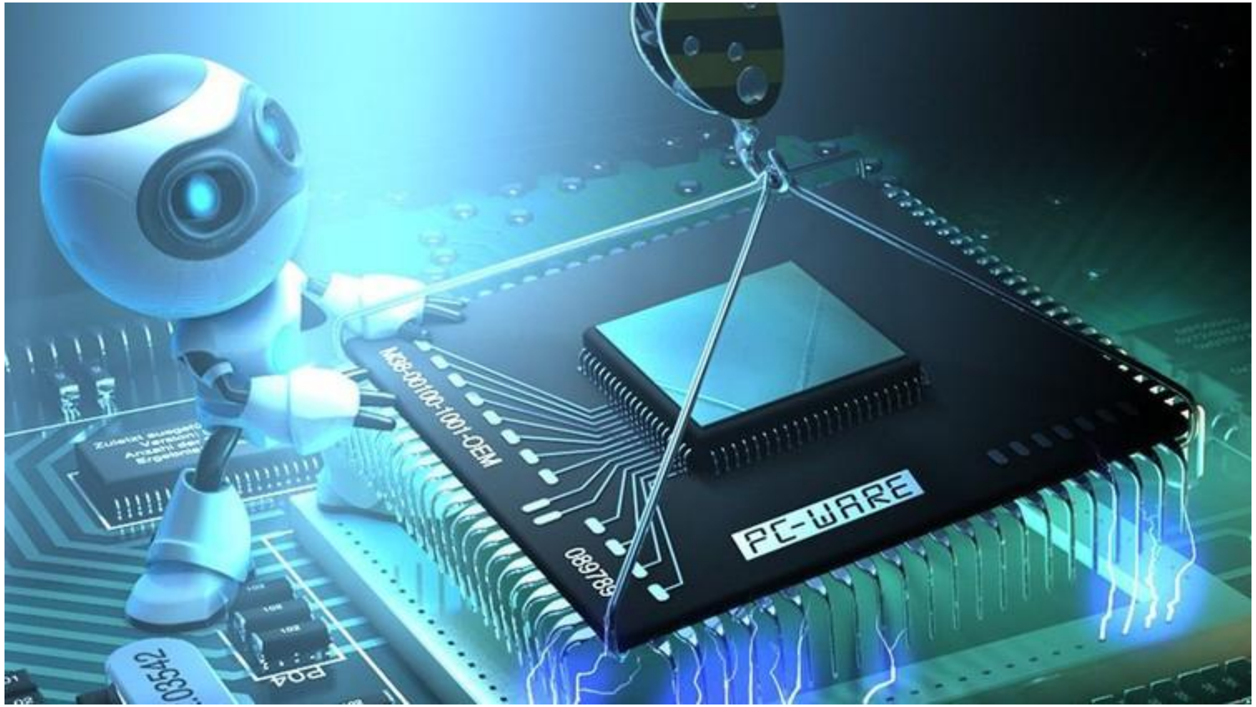
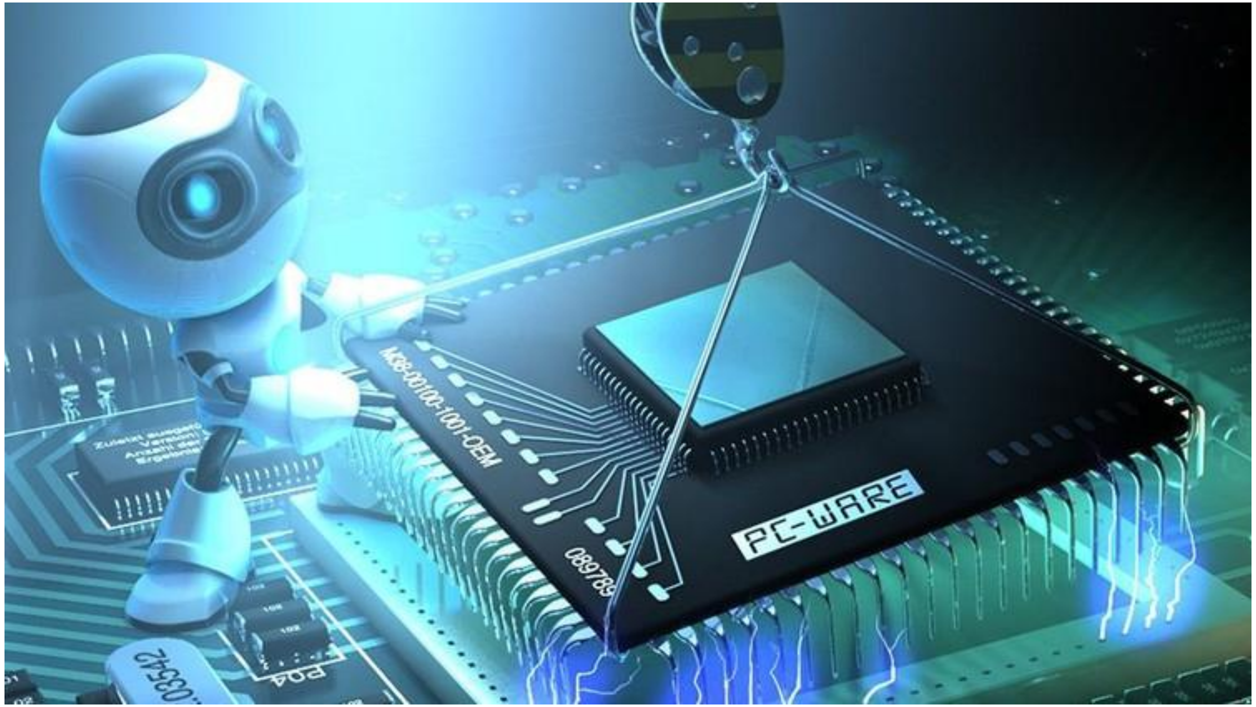
Laboratory Session 2



Practise - Problem 4



Practise - Problem 4

4. Translate the Sort routine to x86 assembler.

```
int Sort(S1 v[]){

  int i, j;

  for (i=0; v[i].k != 0x80000000; i++)
    for (j=i+1; v[j].k != 0x80000000; j++)
      if (v[i].k > v[j].k)
         Swap(v, i, j);

   return i;
}
```

Lucas Bazilio - Udemy

4.

Practise - Problem 4

Part 1/4

```
# Function prologue
pushl %ebp
movl %esp, %ebp

# Allocate space on the stack for local variables
subl $8, %esp  # 4 bytes for i, 4 bytes for j

# Initialize i to 0
movl $0, -4(%ebp)
movl -4(%ebp), %ecx

outer_loop:
  # Check if v[i].k == 0x80000000
  movl 12(%ebp), %eax              #  eax = @v[0].k
  movl (%eax,%ecx,12), %eax        #  eax = v[i].k
  cmpl $0x80000000, %eax
  je done_outer_loop
```

x86

Lucas Bazilio - Udemy

4.

Practise - Problem 4

Part 2/4

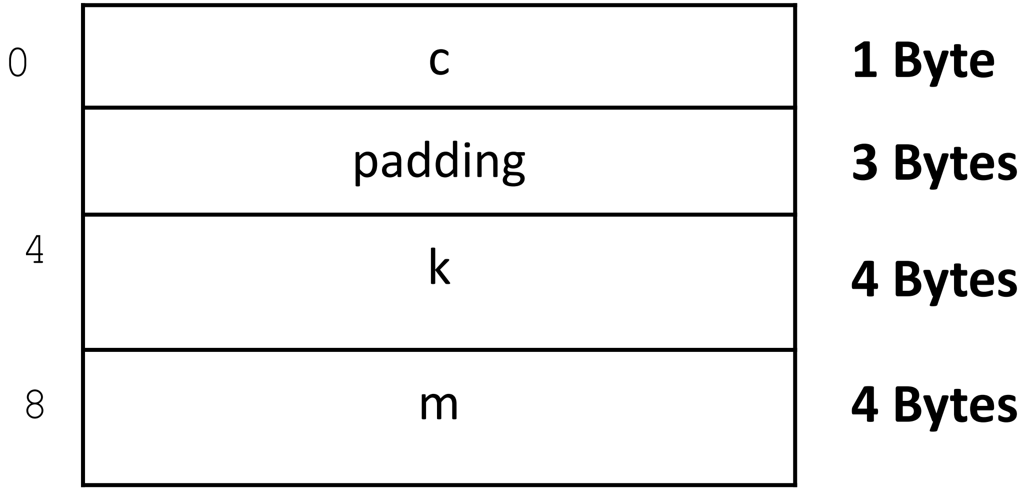
```
# Initialize j to i + 1
movl -4(%ebp), %edx
addl $1, %edx
movl %edx, -8(%ebp)        # j = i + 1

inner_loop:
  # Check if v[j].k == 0x80000000
  movl 12(%ebp), %eax            #  eax = @v[0].k
  movl (%eax,%ebx,12), %eax      #  eax = v[j].k
  cmpl $0x80000000, %eax
  je done_inner_loop

  # Compare v[i].k and v[j].k
  movl 12(%ebp), %eax
  movl (%eax,%ecx,12), %eax            #  eax = v[i].k
  movl 12(%ebp), %edx
  movl (%edx,%ebx,12), %edx            #  edx = v[j].k
  cmpl %edx, %eax
  jge skip_swap
```

x86

Lucas Bazilio - Udemy

Previous Study - Problem 1

```
typedef struct {
   char c;
   int k;
   int *m;
} S1;
```



Total size of struct S1: **12 bytes**

Lucas Bazilio - Udemy

4.

Practise - Problem 4

Part 3/4

```
# Swap v[i] and v[j]
push %ebx
push %ecx
push %eax
call Swap
add $12, %esp

skip_swap:
  # Increment j
  add $1, %ebx
  mov %ebx, -8(%ebp)
  jmp inner_loop
```

x86

Lucas Bazilio - Udemy

4.

Practise - Problem 4

Part 4/4

```
done_inner_loop:
    # Increment i
    add $1, %ecx
    mov %ecx, -4(%ebp)
    jmp outer_loop

done_outer_loop:
  # Load i into %eax and return
  mov -4(%ebp), %eax

  # Function epilogue
  mov %ebp, %esp
  pop %ebp
  ret
```

Lucas Bazilio - Udemy