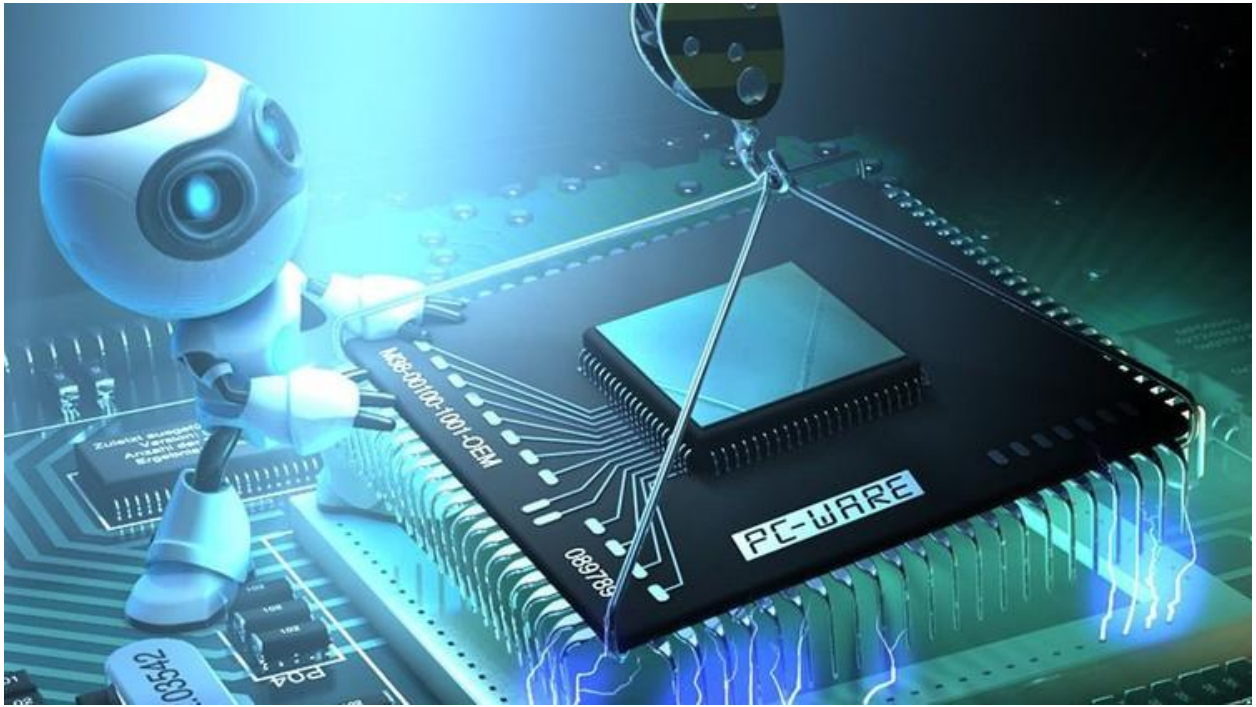
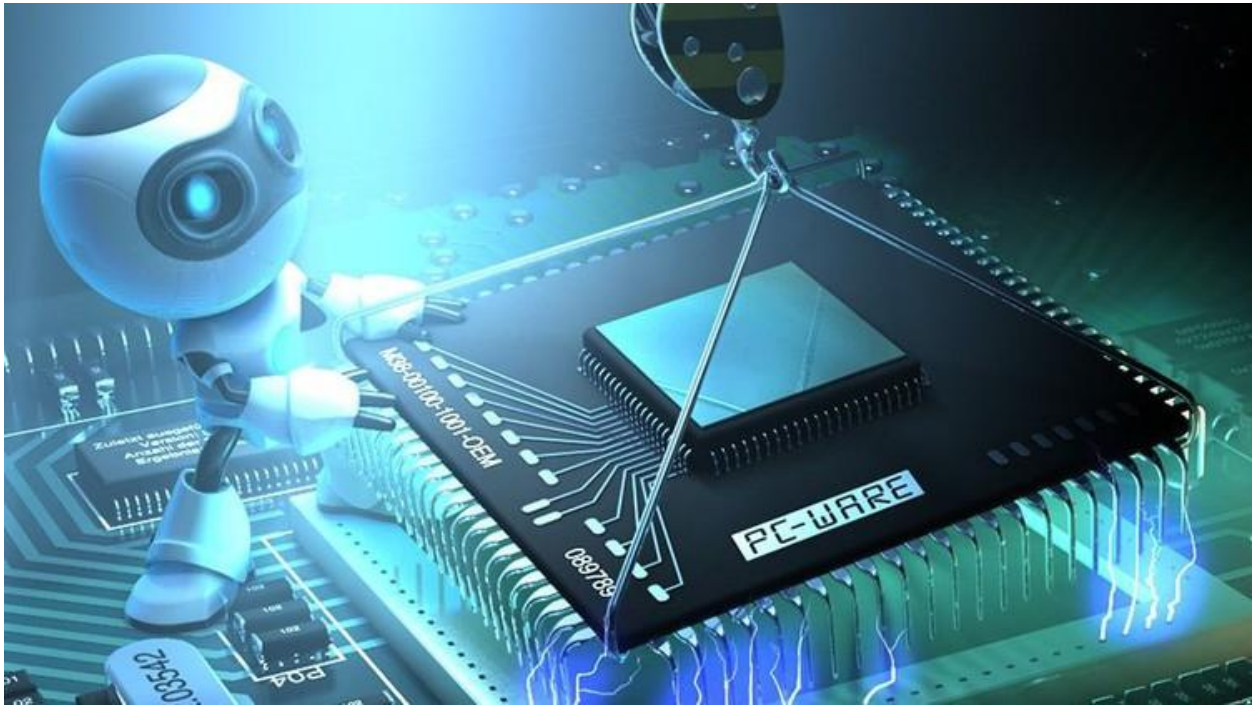
Laboratory Session 2



Practise - Problem 1



Practise - Problem 1

1. Translate the FindElement routine to x86 assembler.

```
int FindElement (int *low, int *high, int *mid, S1 X, S1 v[])
{
        if (X.k == v[*mid].k) { return * mid;  // element found }
        else {
        if (*mid < *high) {
                * mid = * high;
                (* low)++; }
        else {
                * mid = * low;
                (* high)--; }
        }
        return -1;          // element not found
}
```

```
int FindElement (int *low, int *high, int *mid, S1 X, S1 v[])
{
                if (X.k == v[*mid].k) { return * mid;  //      element
found }
                else {
                if (*mid < *high) {
                                * mid = * high;
                                (* low)++; }
                else {
                                * mid = * low;
                                (* high)--; }
                }
                return -1;          // element not found
}
```

Problem 1

1.

Part 1/5

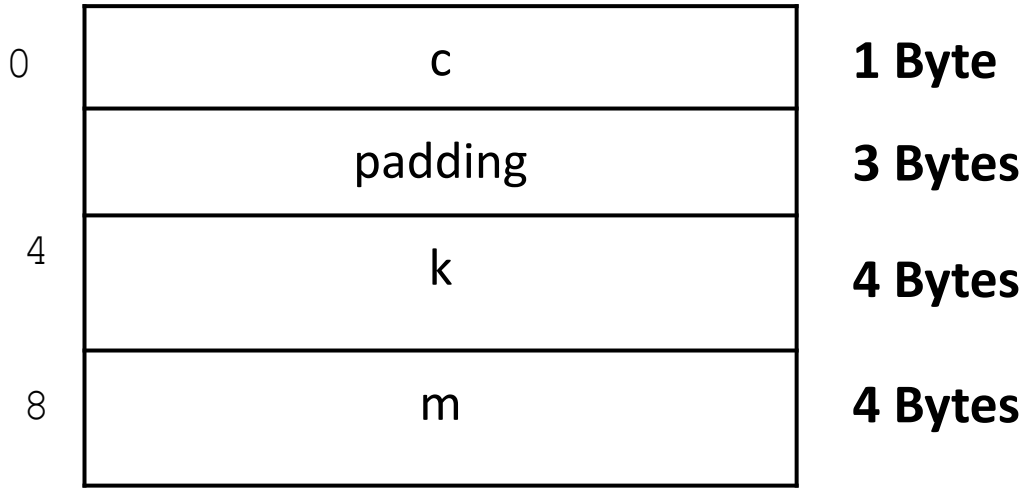
```
.text
  .align 4                                          # align to 4 bytes
  .globl FindElement
  .type FindElement,@function

FindElement:
            pushl       %ebp
            movl %esp, %ebp
            # v[*mid].k
            movl 16(%ebp), %eax                     # %eax = mid (pointer)
            movl (%eax), %eax                       # %eax = *mid (value)
            movl 32(%ebp, %eax, $12), %eax          # eax = @v[*mid].k
            movl (%eax), %eax                       # eax = v[*mid].k
```

x86

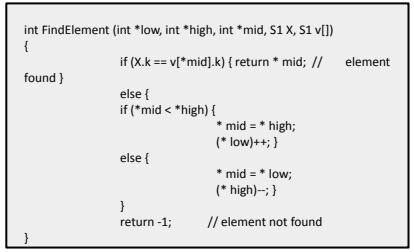
Previous Study - Problem 1

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



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

Problem 1



```
int FindElement (int *low, int *high, int *mid, S1 X, S1 v[])
{
        if (X.k == v[*mid].k) { return * mid;  //      element
found }
        else {
        if (*mid < *high) {
                * mid = * high;
                (* low)++; }
        else {
                * mid = * low;
                (* high)--; }
        }
        return -1;        // element not found
}
```

1.

Part 2/5

```
        # X.k -> 24(%ebp)
        # compare v[*mid].k, X.k
        compl   24(%ebp), %eax
        jne     else1

if1:    # statement if1 -> return * mid
        movl    16(%ebp), %eax
        movl    (%eax), %eax
        jmp endif1
```

x86

Problem 1

1.

```
int FindElement (int *low, int *high, int *mid, S1 X, S1 v[])
{
                if (X.k == v[*mid].k) { return * mid;  //      element
found }
                else {
                if (*mid < *high) {
                                * mid = * high;
                                (* low)++; }
                else {
                                * mid = * low;
                                (* high)--; }
                }
                return -1;          // element not found
}
```

Part 3/5

```
else1:      # statement else1 -> the whole piece of code
            # *mid
            movl 16(%ebp), %eax
            movl (%eax), %eax

            # *high
            movl 12(%ebp), %ecx
            movl (%ecx), %ecx

            # compare *mid, *high
            compl       %ecx, %eax
            jge    else2
```

x86

```
int FindElement (int *low, int *high, int *mid, S1 X, S1 v[])
{
                if (X.k == v[*mid].k) { return * mid;  //      element
found }
                else {
                if (*mid < *high) {
                                * mid = * high;
                                (* low)++; }
                else {
                                * mid = * low;
                                (* high)--; }
                }
                return -1;          // element not found
}
```

Problem 1

1.

Part 4/5

```
if2:        # statement if2 -> * mid = * high; (* low)++;
            movl    16(%ebp), %eax          # &mid
            movl    %ecx, (%eax)            # *mid = * high
            movl    8(%ebp), %edx           # &low
            movl    (%edx), %edx            # *low
            incl  %edx                      # (*low) + 1
            movl    8(%ebp), %ecx           # &low
            movl    %edx, (%ecx)            # *low = (*low) + 1
```

x86

```
int FindElement (int *low, int *high, int *mid, S1 X, S1 v[])
{
                if (X.k == v[*mid].k) { return * mid;  //      element
found }
                else {
                if (*mid < *high) {
                                * mid = * high;
                                (* low)++; }
                else {
                                * mid = * low;
                                (* high)--; }
                }
                return -1;          // element not found
}
```

Problem 1

1.

Part 4/5

```
else2:      # sentence else2 -> * mid = * low; (* high)--;
            movl    8(%ebp), %edx           # &low
            movl    (%edx), %edx            # *low
            movl    16(%ebp), %eax          # &mid
            movl    %edx, (%eax)            # *mid = * low
            decl %ecx                       # (*high) - 1
            movl    12(%ebp), %edx          # &high
            movl    %ecx, (%edx)            # *high = (*high) - 1
```

x86

```
int FindElement (int *low, int *high, int *mid, S1 X, S1 v[])
{
                if (X.k == v[*mid].k) { return * mid;  //      element
found }
                else {
                if (*mid < *high) {
                                * mid = * high;
                                (* low)++; }
                else {
                                * mid = * low;
                                (* high)--; }
                }
                return -1;          // element not found
}
```

Problem 1

1.

Part 5/5

```
endif2:     # statement after else -> return -1;
            movl        $-1, %eax

endif1:
            ret
            movl        %ebp, %esp
            popl %ebp
```

x86