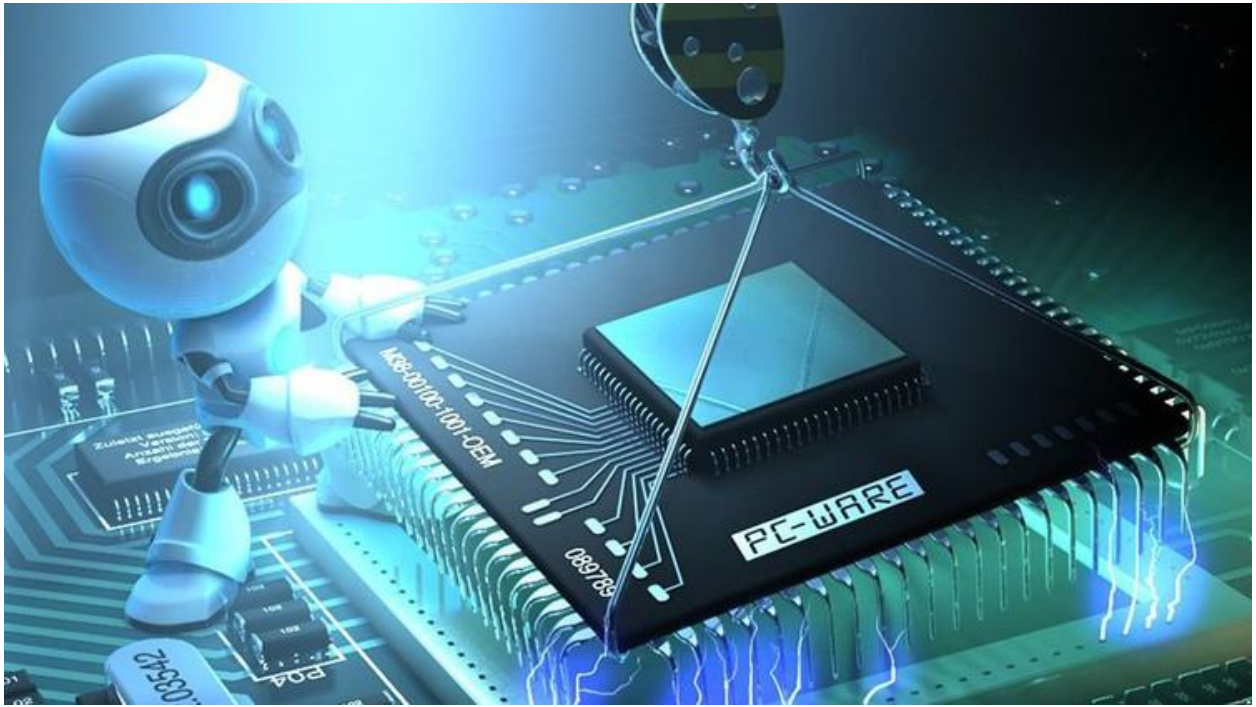
Laboratory Session 1



Objective

The objective of this session is to introduce assembly programming for the x86 architecture. Specifically, aspects such as the programming of control structures (conditional and iterative) and access to structured elements (vectors and matrices) will be worked on.

Tasks

This laboratory session is divided into two tasks:

- Previous Study
- Work to be done during practice

Prior Knowledge

To carry out this practice you should review the direct translations from C to x86 assembler of the control structures that you have seen in the theory lectures. Also you should review the x86 addressing modes.

Prior Knowledge

Vectors

- **Declaration in C:**

```
type name[size];          // indexed starting at 0
```

- **Storage in consecutive memory locations**
  - Access element `V[i]`: **@start V + i*size** (size: size of the elements of V)

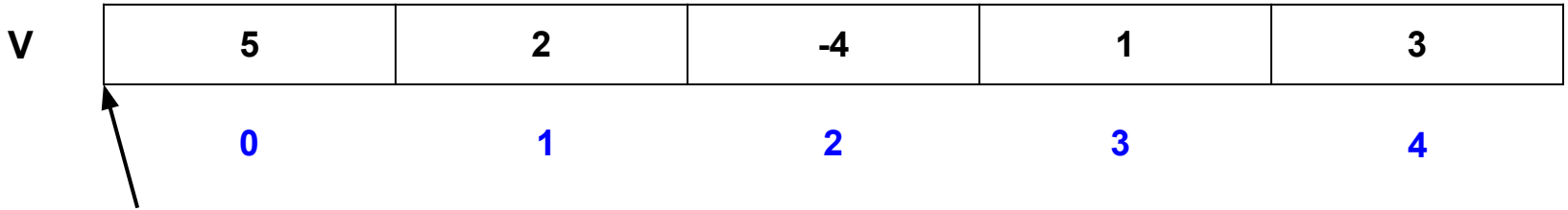
Prior Knowledge

Vectors

- **Declaration in C:**

```
type name[size];          // indexed starting at 0
```

- **Storage in consecutive memory locations**
  - Access element `V[i]`: **@start V + i*size** (size: size of the elements of V)



@V ≡ V[0] **V[3] → @V + 3*4**

Prior Knowledge

Matrices

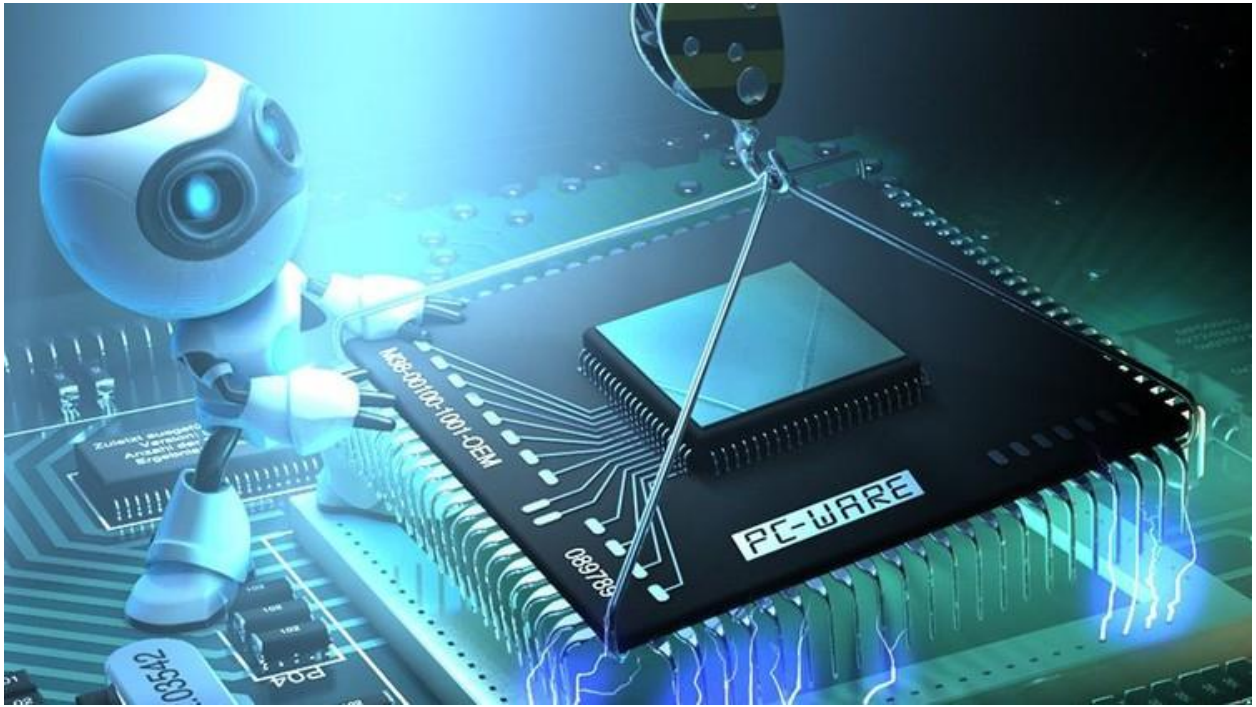
- **Declaration in C:**

```
type name[NumRows][NumColumns];      // indexed starting at (0,0)
```

- **Storage by rows in consecutive memory locations**
  - Access element `A[i][j]`: **@start A + (i*NumColumns + j) * size**

(size: size of the elements of A)

Previous Study



Previous Study

1. Translate the following loop to assembler:

```
#define N 10
int                                          Matrix[N][N],i,sum;

for (i=0,sum=0;i<N;i++)
      sum+=Matrix[3][i];
```

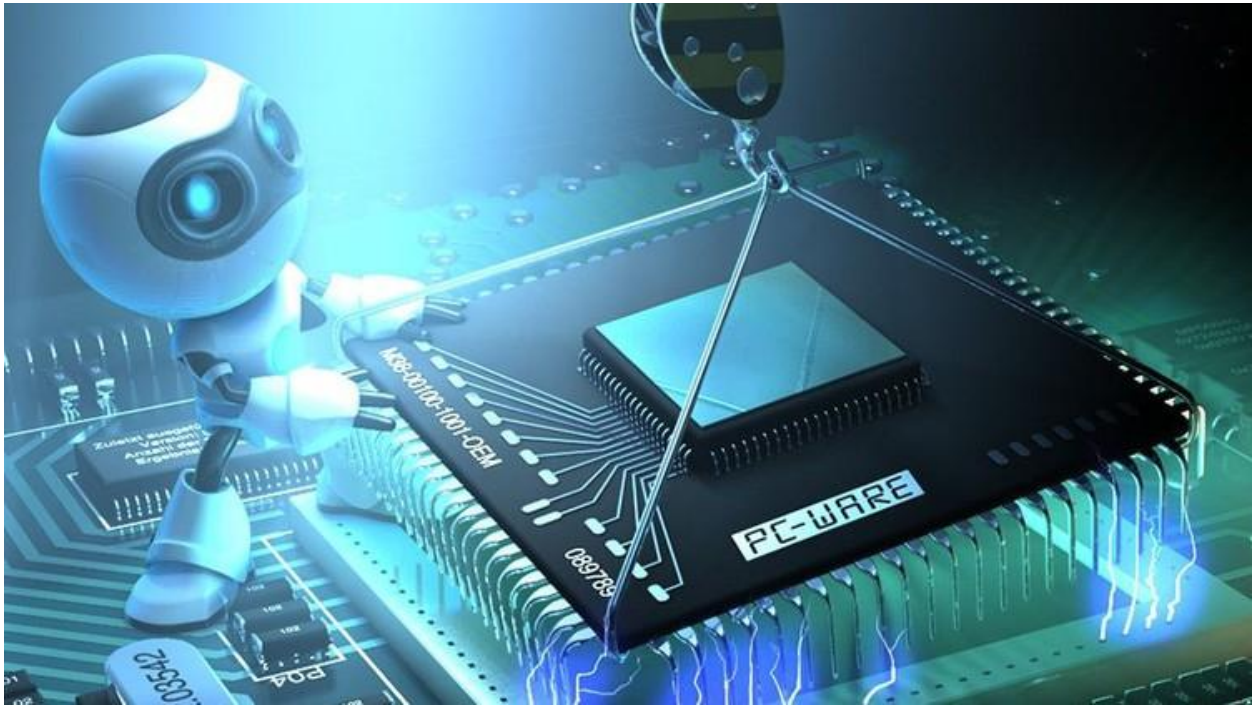
Previous Study

2. Translate the following code to assembler:

```
#define N 10
#define M 100
int Matrix[N][N],i,j,ResRow[N];

for (i=0,j=0,ResRow[0]=1;i<N;i++,j=0,ResRow[i]=1)
  while(Matrix[i][j]!=0) {
    if (Matrix[i][j]==M)
      ResRow[i]*=Matrix[i][j];
    j++;
}
```

Previous Study - Problem 1



Previous Study - Problem 1

1. Translate the following loop to assembler:

```
#define N 10
int                                          Matrix[N][N],i,sum;

for (i=0,sum=0;i<N;i++)
      sum+=Matrix[3][i];
```

Previous Study

1.

```
#define N 10
int         Matrix[N][N],i,sum;

for (i=0,sum=0;i<N;i++)
            sum+=Matrix[3][i];
```

```
        movl $0, %eax     # i = 0
        movl $0, %ebx     # sum = 0
for:    cmpl $10, %eax
        jge endfor
        addl Matrix($120, %eax, $4), %ebx
        incl %eax
        jmp for
endfor:
        movl %ebx, sum
```

x86