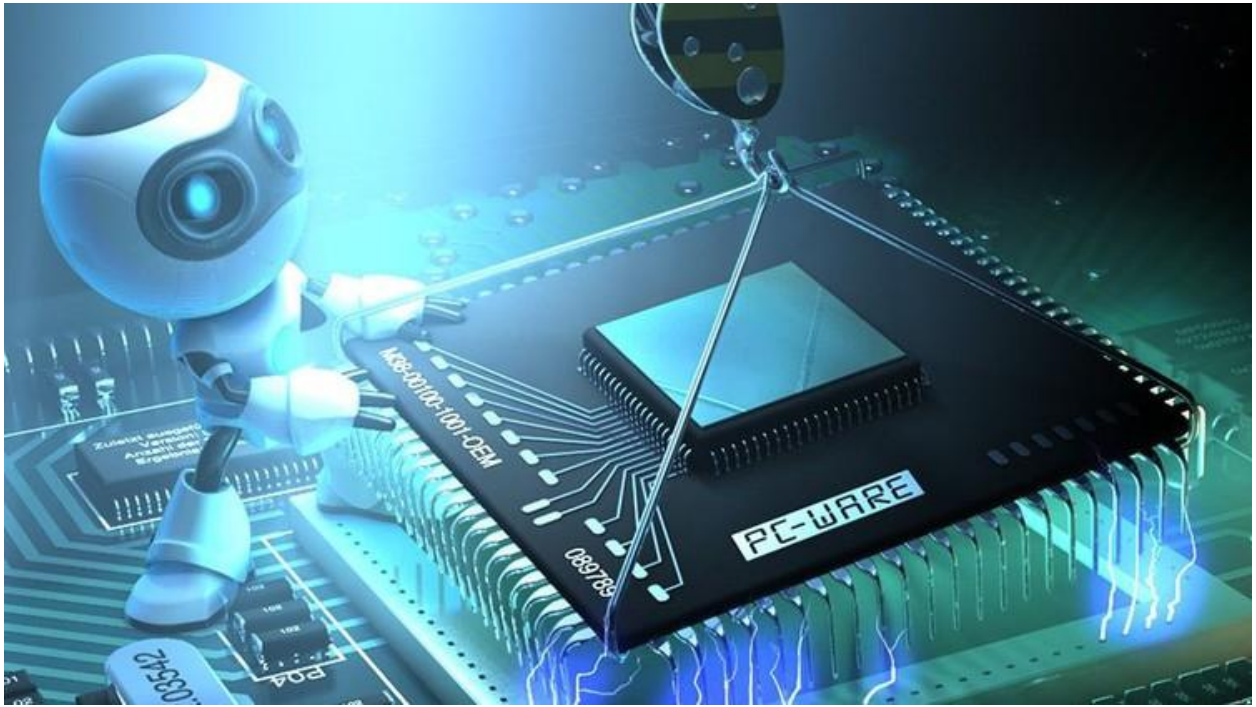
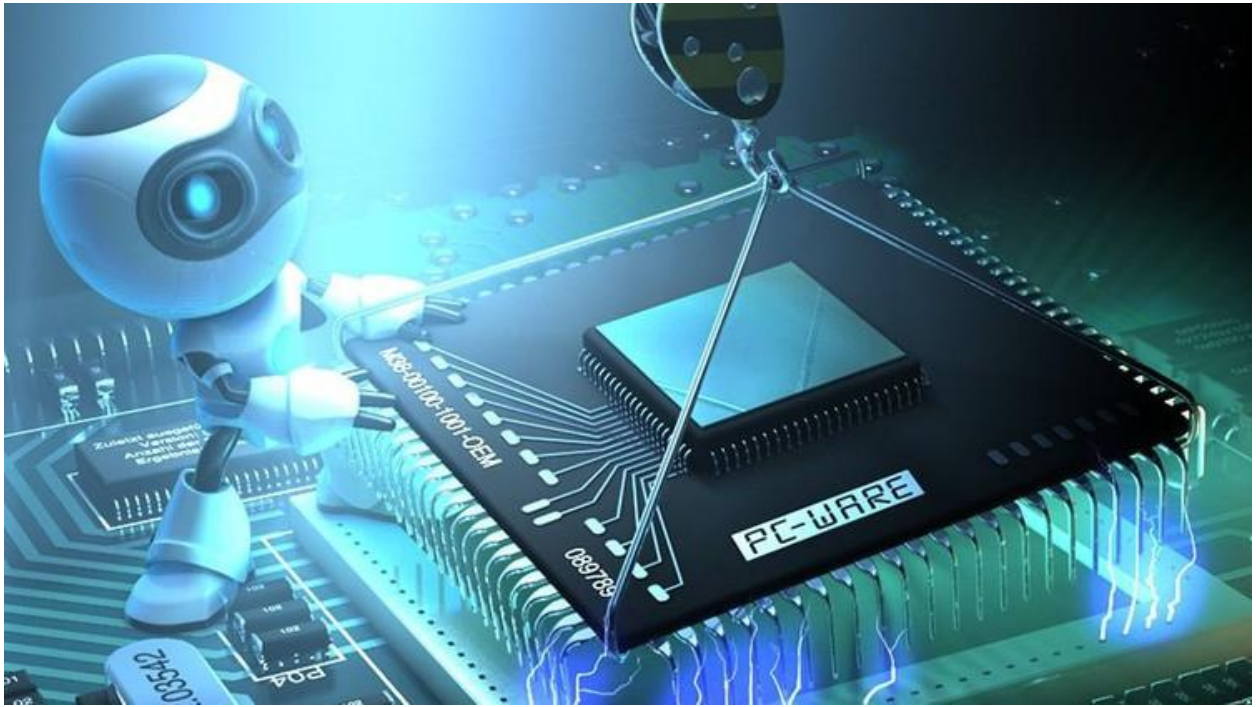
Laboratory Session 1



Previous Study - Problem 2



Iterative Statement (FOR)

MODEL:

```
for (INI; COND; INC) {
    BODY-FOR
}
```

Generic translation:

```
        INI
for:    evaluate condition
        j(fails) end
        BODY-FOR
        INC
        jmp for
end:
```

Previous Study - Problem 2

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

2.

```
#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++;
  }
```

```
                        movl $0, %esi                   # i = 0
for:                    cmpl $10, %esi
                        jge endfor
                        movl $0, %ebx                   # j = 0
                        movl $1, (ResRow, %esi, $4)     # ResRow[i] = 1
while:                  movl (Matrix, %esi, $40), %ecx
                        addl (, %ebx, $4), %ecx         # Matrix[i][j]
                        cmpl $0, %ecx
                        je endwhile

                        cmpl $100, %ecx
                        jne endif

                        movl (ResRow, %esi, 4), %eax
                        imull (%ecx)
                        movl %eax, (ResRow, %esi, 4)

endif:

                        incl %ebx
                        jmp while

endwhile:

                        incl %esi
                        jmp for

endfor:
```

x86

Previous Study

2.

```
#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++;
}
```

Part 1/2

```
                movl $0, %esi                   # i = 0
for:            cmpl $10, %esi
                jge endfor
                movl $0, %ebx                   # j = 0
                movl $1, (ResRow, %esi, $4)     # ResRow[i] = 1
while:          movl (Matrix, %esi, $40), %ecx
                addl (, %ebx, $4), %ecx         # Matrix[i][j]
                cmpl $0, %ecx
                je endwhile
```

x86

Iterative Statement (WHILE)

MODEL:

```
while (cond) {
    BODY-WHILE
}
```

Generic translation:

```
while: evaluate condition
       j(fails) end
       BODY-WHILE
       jmp while
end:
```

Previous Study

2.

```
#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++;
}
```

Part 2/2

```
                cmpl $100, %ecx
                jne endif
                movl (ResRow, %esi, 4), %eax    # ResRow[i]
                imull (%ecx)
                movl %eax, (ResRow, %esi, 4)
endif:
                incl %ebx
                jmp while
endwhile:
                incl %esi
                jmp for
endfor:
```

x86

Lucas Bazilio - Udemy

Advanced Arithmetic Instructions

Instructions	Description	Notes	Example
IMUL op1, op2	op2 ← op2·op1	op2: register	IMUL (%EBX),%EAX
IMUL inm,op1,op2	op2 ← op1·inm	inm: constant	IMUL $3,%EAX,%ECX
IMULL op1	%EDX%EAX ← op1·%EAX	op1: mem. o reg. (**Integers**)	IMULL (%EBX)
MULL op1	%EDX%EAX ← op1·%EAX	op1: mem. o reg. (**Naturals**)	MULL (%EBX)
CLTD	%EDX%EAX ← ExtSign(%EAX)		CLTD
IDIVL op1	%EAX ← %EDX%EAX / op1 %EDX ← %EDX%EAX % op1	op1: mem. o reg. (**Integers**)	IDIVL (%EBX)
DIVL op1	%EAX ← %EDX%EAX / op1 %EDX ← %EDX%EAX % op1	op1: mem. o reg. (**Naturals**)	DIVL %ESI