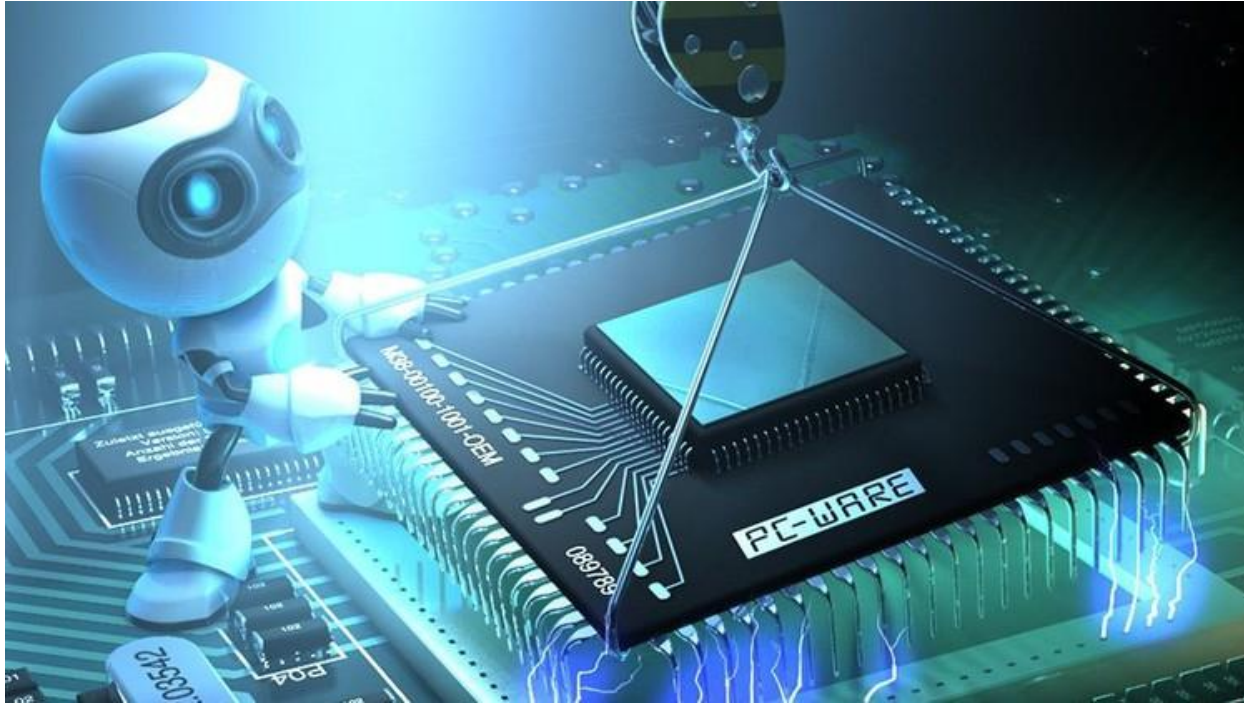


Translation Practise



Structured Data Types



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)

Translation Practise



Example:

```
void Copy (int M[50][80], int X[50][80]) {  
    int i, j;  
    for (i=0; i<50; i++)  
        for (j=0; j<80; j++)  
            M[i][j] = X[i][j];  
}
```

$M[i][j] \rightarrow @M + (i*80 + j) * 4$

$M[i][j] = X[i][j]$

$X[i][j] \rightarrow @X + (i*80 + j) * 4$

$M[i][j] \leftarrow @X + (i*80 + j) * 4$

Translation Practise



Traduction:

```
Copy:  pushl %ebp           # i → %ecx
       movl %esp, %ebp     # j → %edx
       subl 8, %esp
       movl 8(%ebp), %edi   # @M → 8[%ebp]
       movl 12(%ebp), %esi  # @X → 12[%ebp]
       xorl %ecx, %ecx
fori:  cmpl $50, %ecx
       jge endi
       body-FORi
       incl %ecx
       jmp fori
endi:  movl %ebp, %esp
       popl %ebp
       ret
```

```
#body-FORi:
       xorl %edx, %edx
forj:  cmpl $80, %edx
       jge endj
       body-FORj
       incl %edx
       jmp forj
endj:
```

```
#body-FORj:
imull $80, %ecx, %eax
addl %edx, %eax
movl (%esi,%eax,4), %ebx
movl %ebx, (%edi,%eax,4)
```

$80*i$
 $80*i + j$
 $\%ebx \leftarrow X[i][j]$

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:
```

Advanced Arithmetic Instructions



Instructions	Description	Notes	Example
IMUL op1, op2	$op2 \leftarrow op2 \cdot op1$	op2: register	IMUL (%EBX),%EAX
IMUL inm,op1,op2	$op2 \leftarrow op1 \cdot inm$	inm: constant	IMUL \$3,%EAX,%ECX
IMULL op1	$\%EDX\%EAX \leftarrow op1 \cdot \%EAX$	op1: mem. o reg. (Integers)	IMULL (%EBX)
MULL op1	$\%EDX\%EAX \leftarrow op1 \cdot \%EAX$	op1: mem. o reg. (Naturals)	MULL (%EBX)
CLTD	$\%EDX\%EAX \leftarrow \text{ExtSign}(\%EAX)$		CLTD
IDIVL op1	$\%EAX \leftarrow \%EDX\%EAX / op1$ $\%EDX \leftarrow \%EDX\%EAX \% op1$	op1: mem. o reg. (Integers)	IDIVL (%EBX)
DIVL op1	$\%EAX \leftarrow \%EDX\%EAX / op1$ $\%EDX \leftarrow \%EDX\%EAX \% op1$	op1: mem. o reg. (Naturals)	DIVL %ESI