Algorithm with Assembly Language

Assembly Languages

- One step up from machine language
- Originally a more user-friendly way to program
- Now mostly a compiler target
- Model of computation: stored program computer

Assembly Language Instructions

- Built from two pieces
- Add AL, [SI]

Types of Opcodes

- Arithmetic, logical
  - add, sub, mul
  - and, or
  - cmp
- Memory load/store
  - mov
- Control transfer
  - jmp
  - jne
- Complex
  - movs

ENIAC, 1946
17k tubes, 5kHz
Operands

- Each operand taken from a particular addressing mode:
- Examples:
  - Register: ADD AL, BL,
  - Immediate: ADD AH, 10
  - Indirect: MOV AL, [SI]
  - PC Relative: JA 100

---

Program 2

- Write an assembly program that will prompt the user to enter an integer \( n \) in the range 1 to 9 (e.g. 7) and produce \( n \) number of lines of output as follows:
  - \( 7777777 \)
  - \( 666666 \) \( 55555 \)
  - \( 4444 \) \( 333221 \)

---

i386 Programmer’s Model

<table>
<thead>
<tr>
<th>Index</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>eip</td>
</tr>
<tr>
<td>1-15</td>
<td>eflags</td>
</tr>
<tr>
<td>16</td>
<td>ebp</td>
</tr>
<tr>
<td>17</td>
<td>esp</td>
</tr>
<tr>
<td>18</td>
<td>edi</td>
</tr>
<tr>
<td>19</td>
<td>esi</td>
</tr>
<tr>
<td>20</td>
<td>ebx</td>
</tr>
<tr>
<td>21</td>
<td>edx</td>
</tr>
<tr>
<td>22</td>
<td>ecx</td>
</tr>
<tr>
<td>23-31</td>
<td>eax</td>
</tr>
</tbody>
</table>

---

Title Program 3

```
.model small
.stack 100h
.data
    count byte ?
.code
    ;data segment
    ;code segment
    ;proc declares the start of a proc
Main proc
    mov ax, @data ;copy the addr. of data
    mov ds, ax ;segment into DS register
```
Program 6

- Write and test an assembly program which takes an integer argument and returns an integer result. This function $f(x)$ shall return the value:
  - $f(x) = 1$ if $x = 0$; and
  - $f(x) = x \times (x-1) \times (x-2) \times \ldots \times 2 \times 1$; if $x$ is greater than 1.
- Test your function with the argument 5.
Simple Searching

- To find the largest integer in an array

\{ 5, 20, 1, 8, 9, 50, 60, 87, 89, 45\};

Title searching algorithm

```
.model small
.stack 100h
.data segment
.data
input byte 5, 20, 1, 8, 9, 50, 60, 87, 89, 45 ;integer array
max word ?
len byte 10 ;length of array
.code segment
.code
Main proc ;proc declares the start of a proc
mov ax, @data ;copy the addr. of data
mov ds, ax ;segment into DS register
mov ax, 1
L1: mul input ;multiplies AX by a source
dec input ;operand
jnz L1
mov ax, 4c00h ;halt the program and
int 21h ;return control to DOS
Main ENDP ;marks the end of current procedure
END Main ;the program exit point
```
Displaying results

- To create a print function to display the largest number;
- Note that we need to convert the result to corresponding ASCII codes in order to display on the console;

```
Print_proc
push dx
    mov ah, 9
    mov dx, offset message
    int 21h
pop dx
    mov ah, 6 ; display the most significant byte
    mov dl, dh ; of the variable.
    and dl, 0f0h ; use AND operator to mask off the least
    shr dl, 1 ; significant byte
    shr dl, 1 ; shift right 4 times
    shr dl, 1
    shr dl, 1
    add dl, 48 ; corresponding ASCII code
    int 21h ; call DOS service
Print ENDP
```

Sorting Algorithm

- To arrange the previous array in the ascending order

```
mov ah, 6
    mov dl, dh
    and dl, 00fh
    add dl, 48
    int 21h

mov ah, 9
    mov dx, offset newline
    int 21h
ret
Print ENDP
```
Title: Sorting algorithm

.model small
.stack 100h

.data
array byte 5, 20, 1, 8, 9, 50, 60, 87, 89, 45
len word 10

.code
Main proc ;proc declares the start of a proc
mov ax, @data ;copy the addr of data
mov ds, ax ;segment into DS register
mov si, 0

L1: mov bh, [array+si] ;starting pointer
mov di, si ;pointer to subsequent elements
L2: cmp bh, [array+di] ;jump to L3 if the order is ascending
    jna L3 ;otherwise, swap two elements
    mov bl, [array+di]
    mov bh, [array+si]
    xchg bh, bl
    mov [array+di], bl
    mov [array+si], bh
L3: inc di
    cmp di, len
    jne L2
    inc si
    cmp si, len
    jne L1

mov ax, 4c00h ;halt the program and
int 21h ;return control to DOS
Main ENDP ;marks the end of current procedure
END Main ;the program exit point