3.1 Software: The Microcomputer Program

An instruction can be divided into two parts:
- Operation code (opcode) – one- to five-letter mnemonic
- Operands

Format of an assembly statement:

LABEL: INSTRUCTION ; COMMENT

e.g. START: MOV AX, BX ; Copy BX into AX

3.2 Assembly Language Programming

Development on the PC

3.3 The Instruction Set

3.4 The MOV Instruction

3.5 Addressing Modes

3.1 Software: The Microcomputer Program

The sequence of commands used to tell a microcomputer what to do is called a program. Each command in a program is an instruction.

The 8088 microprocessor performs operations for 117 basic instructions.

Programs must be coded in machine language before they can be run by the 8088. However, programs are normally written in 8088 assembly language or a high-level language such as C.

A single machine language instruction can take one to six bytes of code.
3.1 Software: The Microcomputer Program

- Assembly language must be converted by an assembler to an equivalent machine language program for execution by the 8088.
- A directive is a statement that is used to control the translation process of the assembler.
  e.g. D8 64 DUP(?)
- The machine language output produced by the assembler is called object code.
- Listing of the assembled program
  e.g. 0013 8A 24 NXTPT: MOV AH, [SI] ; Move a byte

3.2 Assembly Language Programming Development on the PC

- Describing the problem
- Planning the solution
- Coding the solution with assembly language
- Creating the source program
- Assembling the source program into an object module
- Producing a run module
- Verifying the solution
- Programs and files involved in the program development cycle
3.2 Assembly Language Programming

Development on the PC

Program development cycle

Given problem
Describe problem
Plan steps of solution
Flowchart
Implement flowchart using assembler language
Enter/edit source program using the editor
Assemble the program using the assembler

Begin
Begin block move

Execute
Move an element from source to destination block

No
All elements moved?
Yes
End

End block move

Syntax errors?
Yes
Logic errors?
No
Solution to problem

Input/output
Process
Decision

Commonly used flowchart symbols

Subroutine
Connection within a flowchart
Connection to another flowchart

Coding the solution with assembly language
Two types of statements are used in the source program
- The assembly language instructions
- The directives

The assembly language instructions are used to tell the microprocessor what operations are to be performed to implement the application.

A directive is the instruction to the assembler program used to convert the assembly language program into machine code.

Describing the problem
Most applications are described with a written document called an application specification.

Planning the solution
A flow chart is an outline that both documents the operations that must be performed by software to implement the planned solution and shows the sequence in which they are performed.

Flow chart of a block-move program

Describing the problem

Most applications are described with a written document called an application specification.

Planning the solution

A flow chart is an outline that both documents the operations that must be performed by software to implement the planned solution and shows the sequence in which they are performed.

Flow chart of a block-move program
3.2 Assembly Language Programming

Development on the PC

- Coding the solution with assembly language
  - The assembly language instructions
    - [Example]
      MOV AX, DATASEGMENT
      MOV DS, AX
      MOV SI, BLK1ADDR
      MOV DI, BLK2ADDR
      MOV CX, N
  - The directive
    - [Example]
      BLOCK PROC    FAR
      or BLOCK ENDP

- Creating the source program
  - The EDIT editor
  - The Notepad editor in Windows
  - The Microsoft PWB (Programmer’s Work Bench)

- Assembling the source program into an object module
  - The Microsoft MASM assembler
  - The Microsoft PWB (Programmer’s Work Bench)
  - The assembler source file and the object module

- Producing a run module
  - The object module must be processed by the LINK program to produce an executable run module.

- Verifying the solution
  - Programs and files involved in the program development cycle
    - PROG1.ASM (Editor)
    - PROG1.OBJ (Assembler)
    - PROG1.LST (Assembler)
    - PROG1.EXE (Linker)
    - PROG1.MAP (Linker)

3.3 The Instruction Set

- The instruction set of a microprocessor defines the basic operations that a programmer can specify to the device to perform.
- Instruction set groups
  - Data transfer instructions
  - Arithmetic instructions
  - Logic instructions
  - String manipulation instructions
  - Control transfer instructions
  - Processor control instructions

- Data transfer instructions

---
3.3 The Instruction Set

- Data transfer instructions

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Instruction Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOV</td>
<td>00011000</td>
</tr>
<tr>
<td>ADD</td>
<td>00011011</td>
</tr>
<tr>
<td>SUB</td>
<td>00011010</td>
</tr>
<tr>
<td>AND</td>
<td>00011001</td>
</tr>
<tr>
<td>ORR</td>
<td>00011010</td>
</tr>
<tr>
<td>MOVK</td>
<td>00011001</td>
</tr>
<tr>
<td>ADDK</td>
<td>00011010</td>
</tr>
<tr>
<td>SUBK</td>
<td>00011010</td>
</tr>
<tr>
<td>ANDK</td>
<td>00011001</td>
</tr>
<tr>
<td>ORRK</td>
<td>00011010</td>
</tr>
</tbody>
</table>

- Arithmetic instructions

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Instruction Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>00011011</td>
</tr>
<tr>
<td>SUB</td>
<td>00011010</td>
</tr>
<tr>
<td>AND</td>
<td>00011001</td>
</tr>
<tr>
<td>ORR</td>
<td>00011010</td>
</tr>
<tr>
<td>MOVK</td>
<td>00011001</td>
</tr>
<tr>
<td>ADDK</td>
<td>00011010</td>
</tr>
<tr>
<td>SUBK</td>
<td>00011010</td>
</tr>
<tr>
<td>ANDK</td>
<td>00011001</td>
</tr>
<tr>
<td>ORRK</td>
<td>00011010</td>
</tr>
</tbody>
</table>

- Logic instructions

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Instruction Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOV</td>
<td>00011000</td>
</tr>
<tr>
<td>ADD</td>
<td>00011011</td>
</tr>
<tr>
<td>SUB</td>
<td>00011010</td>
</tr>
<tr>
<td>AND</td>
<td>00011001</td>
</tr>
<tr>
<td>ORR</td>
<td>00011010</td>
</tr>
<tr>
<td>MOVK</td>
<td>00011001</td>
</tr>
<tr>
<td>ADDK</td>
<td>00011010</td>
</tr>
<tr>
<td>SUBK</td>
<td>00011010</td>
</tr>
<tr>
<td>ANDK</td>
<td>00011001</td>
</tr>
<tr>
<td>ORRK</td>
<td>00011010</td>
</tr>
</tbody>
</table>

- String manipulation instructions

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Instruction Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOV</td>
<td>00011000</td>
</tr>
<tr>
<td>ADD</td>
<td>00011011</td>
</tr>
<tr>
<td>SUB</td>
<td>00011010</td>
</tr>
<tr>
<td>AND</td>
<td>00011001</td>
</tr>
<tr>
<td>ORR</td>
<td>00011010</td>
</tr>
<tr>
<td>MOVK</td>
<td>00011001</td>
</tr>
<tr>
<td>ADDK</td>
<td>00011010</td>
</tr>
<tr>
<td>SUBK</td>
<td>00011010</td>
</tr>
<tr>
<td>ANDK</td>
<td>00011001</td>
</tr>
<tr>
<td>ORRK</td>
<td>00011010</td>
</tr>
</tbody>
</table>

- Control transfer instructions

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Instruction Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOV</td>
<td>00011000</td>
</tr>
<tr>
<td>ADD</td>
<td>00011011</td>
</tr>
<tr>
<td>SUB</td>
<td>00011010</td>
</tr>
<tr>
<td>AND</td>
<td>00011001</td>
</tr>
<tr>
<td>ORR</td>
<td>00011010</td>
</tr>
<tr>
<td>MOVK</td>
<td>00011001</td>
</tr>
<tr>
<td>ADDK</td>
<td>00011010</td>
</tr>
<tr>
<td>SUBK</td>
<td>00011010</td>
</tr>
<tr>
<td>ANDK</td>
<td>00011001</td>
</tr>
<tr>
<td>ORRK</td>
<td>00011010</td>
</tr>
</tbody>
</table>
3.3 The Instruction Set

- Control transfer instructions

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Meaning</th>
<th>Format</th>
<th>Operation</th>
<th>Flags affected</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOV</td>
<td>Move</td>
<td>MOV D, S</td>
<td>(B) → (D)</td>
<td>None</td>
</tr>
</tbody>
</table>

e.g.: MOV DX, CS

MOV [SUM], AX

3.4 The MOV Instruction

- The move (MOV) instruction is used to transfer a byte or a word of data from a source operand to a destination operand.

- Process control instructions

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Meaning</th>
<th>Format</th>
<th>Operation</th>
<th>Flags affected</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOV</td>
<td>Move</td>
<td>MOV D, S</td>
<td>(B) → (D)</td>
<td>None</td>
</tr>
</tbody>
</table>

e.g.: MOV DX, CS

MOV [SUM], AX
3.4 The MOV Instruction

- MOV DX, CS

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Memory Location</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOV DX, CS</td>
<td>0100</td>
<td>XX</td>
</tr>
</tbody>
</table>

8088/8086 MPU

3.5 Addressing Modes

- Register operand addressing mode
- Immediate operand addressing mode
- Memory operand addressing mode
  - Direct addressing mode
  - Register indirect addressing mode
  - Based addressing mode
  - Indexed addressing mode
  - Based-indexed addressing mode

3.5 Addressing Modes

- Register operand addressing mode

Before execution:

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Memory Location</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOV AX, BX</td>
<td>0100</td>
<td>XX</td>
</tr>
</tbody>
</table>

After execution:

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Memory Location</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOV AX, BX</td>
<td>0100</td>
<td>XX</td>
</tr>
</tbody>
</table>

3.5 Addressing Modes

- Immediate operand addressing mode

Before execution:

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Memory Location</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOV AL, 15H</td>
<td>0100</td>
<td>XX</td>
</tr>
</tbody>
</table>

After execution:

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Memory Location</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOV AL, 15H</td>
<td>0100</td>
<td>XX</td>
</tr>
</tbody>
</table>
### 3.5 Addressing Modes

#### Immediate operand addressing mode

- **Address**
- **Memory Content**
- **Instruction**

```
MOV AL, 15H
```

`0100000015`

**Example:** MOV AX, [1234H]

---

#### Direct addressing mode

- **Address**
- **Memory Content**
- **Instruction**

```
MOV CX, [1234H]
```

**Example:** MOV AX, [SI]

---

#### Register indirect addressing mode

- **Address**
- **Memory Content**
- **Instruction**

```
MOV AL, 15H
```

**Example:** MOV AX, [SI]
3.5 Addressing Modes

- **Register indirect addressing mode**
  \[ PA = 0200_{16} + 1234_{16} = 03234_{16} \]

- **Indexed addressing mode**
  \[ PA = 0200_{16} + 1000_{16} + 1234_{16} = 04234_{16} \]

- **Based addressing mode**
  \[ PA = \text{Segment Base} : \text{Index} + \text{Displacement} \]

  - Example: MOV [BX]+1234H, AL
  - Computation of a based address:
    \[ \text{PA} = (\text{BX} + 1234_{16}) \]
  - Computation of an indexed address:
    \[ \text{PA} = (\text{SI} + 1234_{16}) \]

---

Before execution

After execution
3.5 Addressing Modes

- Memory addressing modes - Indexed addressing mode

PA = \(O2000_{16} + 2000_{16} + 1234_{16} = 05234_{16}\)

8088/8086 MPU

Address | Memory Content | Instruction
--- | --- | ---
01000 | XX | Next Instruction
01001 | XX | Next Instruction
01002 | XX | Next Instruction
01003 | XX | Next Instruction
01004 | XX | Next Instruction
02000 | XX | Source operand
02001 | XX | Source operand

3.5 Addressing Modes

Memory addressing modes - Based-indexed addressing mode

PA = \(O2000_{16} + 1000_{16} + 2000_{16} + 1234_{16} = 06234_{16}\)

8088/8086 MPU

Address | Memory Content | Instruction
--- | --- | ---
01000 | XX | Next Instruction
01001 | XX | Next Instruction
01002 | XX | Next Instruction
01003 | XX | Next Instruction
01004 | XX | Next Instruction
02000 | XX | Source operand
02001 | XX | Source operand

3.5 Addressing Modes

Memory addressing modes - Based-indexed addressing mode

PA = \(O2000_{16} + 1000_{16} + 2000_{16} + 1234_{16} = 06234_{16}\)

8088/8086 MPU

Address | Memory Content | Instruction
--- | --- | ---
01000 | XX | Next Instruction
01001 | XX | Next Instruction
01002 | XX | Next Instruction
01003 | XX | Next Instruction
01004 | XX | Next Instruction
02000 | XX | Source operand
02001 | XX | Source operand

3.5 Addressing Modes

Memory addressing modes - Based-indexed addressing mode

PA = \(O2000_{16} + 1000_{16} + 2000_{16} + 1234_{16} = 06234_{16}\)

8088/8086 MPU

Address | Memory Content | Instruction
--- | --- | ---
01000 | XX | Next Instruction
01001 | XX | Next Instruction
01002 | XX | Next Instruction
01003 | XX | Next Instruction
01004 | XX | Next Instruction
02000 | XX | Source operand
02001 | XX | Source operand

3.5 Addressing Modes

Memory addressing modes - Based-indexed addressing mode

PA = \(O2000_{16} + 1000_{16} + 2000_{16} + 1234_{16} = 06234_{16}\)

8088/8086 MPU

Address | Memory Content | Instruction
--- | --- | ---
01000 | XX | Next Instruction
01001 | XX | Next Instruction
01002 | XX | Next Instruction
01003 | XX | Next Instruction
01004 | XX | Next Instruction
02000 | XX | Source operand
02001 | XX | Source operand

3.5 Addressing Modes

Memory addressing modes - Based-indexed addressing mode

PA = \(O2000_{16} + 1000_{16} + 2000_{16} + 1234_{16} = 06234_{16}\)

8088/8086 MPU

Address | Memory Content | Instruction
--- | --- | ---
01000 | XX | Next Instruction
01001 | XX | Next Instruction
01002 | XX | Next Instruction
01003 | XX | Next Instruction
01004 | XX | Next Instruction
02000 | XX | Source operand
02001 | XX | Source operand