

**🔹 (a) 16-bit Instruction Mode**

এই ফরম্যাটটি 8086, 80186, 80286 প্রসেসরে ব্যবহৃত হয়।  
এখানে প্রতিটি নির্দেশনা (instruction) ৪টি মূল অংশে বিভক্ত:

1. **Opcode (1–2 bytes)**  
   👉 নির্দেশ করে কী কাজ করতে হবে (যেমন MOV, ADD ইত্যাদি)।
2. **MOD-REG-R/M (0–1 bytes)**  
   👉 কোন রেজিস্টার বা মেমোরি লোকেশনে কাজ হবে, তা নির্দেশ করে।
3. **Displacement (0–1 bytes)**  
   👉 যদি মেমোরি অ্যাড্রেসের সঙ্গে অফসেট প্রয়োজন হয়, তাহলে সেটা এখানে থাকে।
4. **Immediate (0–2 bytes)**  
   👉 যদি কোনো নির্দিষ্ট মান সরাসরি নির্দেশনায় থাকে (যেমন MVI A, 05H), তা এখানে সংরক্ষিত হয়।

**🔹 (b) 32-bit Instruction Mode**

এটি **80386 থেকে Pentium 4** পর্যন্ত ব্যবহৃত হয়, যেখানে ৩২-বিট ডেটা ও অ্যাড্রেস ব্যবহার করা যায়।  
এই ফরম্যাটটি কিছুটা বড় ও জটিল, কারণ এটি আরও বেশি অপশন দেয়।

1. **Address size (0–1 bytes)**  
   👉 ঠিকানার আকার (16-bit বা 32-bit) নির্ধারণ করে।
2. **Register size (0–1 bytes)**  
   👉 রেজিস্টারের আকার নির্ধারণ করে (যেমন 8-bit, 16-bit, বা 32-bit)।
3. **Opcode (1–2 bytes)**  
   👉 কোন কাজটি সম্পন্ন হবে তা নির্দেশ করে।
4. **MOD-REG-R/M (0–1 bytes)**  
   👉 কোন রেজিস্টার বা মেমোরি ব্যবহৃত হবে, তা নির্দেশ করে।
5. **Scaled-index (0–1 bytes)**  
   👉 ইনডেক্স রেজিস্টার ব্যবহার করা হলে, স্কেল (গুণন) ফ্যাক্টর নির্ধারণ করে।
6. **Displacement (0–4 bytes)**  
   👉 মেমোরি ঠিকানার অফসেট নির্দেশ করে।
7. **Immediate (0–4 bytes)**  
   👉 নির্দেশনায় সরাসরি মান থাকলে, সেটি এখানে সংরক্ষিত হয়।

**🧭 সারসংক্ষেপ (Figure 4–1 এর ব্যাখ্যা)**

* (a) অংশে: 16-bit নির্দেশনার গঠন দেখানো হয়েছে (8086 ভিত্তিক প্রসেসরের জন্য)।
* (b) অংশে: 32-bit নির্দেশনার গঠন দেখানো হয়েছে (80386 থেকে Pentium 4 পর্যন্ত)।
* উভয় ক্ষেত্রেই “Opcode” অংশটি নির্দেশনার **মূল কাজ নির্ধারণ করে**, এবং বাকি অংশগুলো ঠিক করে **কোথায় ও কীভাবে কাজটি হবে**।

**MOD field** হলো **MOD-REG-R/M** বাইটের একটি অংশ, যা **Intel 8086 (এবং পরবর্তী)** মাইক্রোপ্রসেসরগুলিতে ব্যবহৃত হয়।  
এই বাইটটি নির্দেশ করে — **অপারেন্ড (operand)** কোথা থেকে আসবে (মেমোরি নাকি রেজিস্টার থেকে), এবং যদি মেমোরি থেকে হয় তবে **কীভাবে তার ঠিকানা (address)** নির্ধারণ হবে।

## 🔹 MOD-REG-R/M Byte কী?

MOD-REG-R/M হলো একটি **৮-বিট ফিল্ড**, যা ৩টি অংশে বিভক্ত:

| **Field** | **Bit Position** | **কাজ** |
| --- | --- | --- |
| **MOD** | Bit 7–6 | অ্যাড্রেসিং মোড নির্দেশ করে (মেমোরি নাকি রেজিস্টার) |
| **REG** | Bit 5–3 | কোন রেজিস্টার ব্যবহার হবে তা নির্দেশ করে |
| **R/M** | Bit 2–0 | কোন রেজিস্টার বা মেমোরি লোকেশন ব্যবহার হবে তা নির্দেশ করে |

## 🔸 MOD Field এর কাজ

**MOD field (2 bits)** নির্ধারণ করে **addressing mode** — অর্থাৎ, ডেটা কোথা থেকে আসবে।

| **MOD Bits** | **Addressing Type** | **অর্থ** |
| --- | --- | --- |
| **00** | Memory (no displacement) | শুধুমাত্র মেমোরি ঠিকানা, কোনো অতিরিক্ত অফসেট নেই। |
| **01** | Memory (8-bit displacement) | মেমোরি ঠিকানার সাথে ৮-বিট অফসেট যোগ হবে। |
| 10 | Memory (16-bit displacement) | মেমোরি ঠিকানার সাথে ১৬-বিট অফসেট যোগ হবে। |
| **11** | Register mode | উভয় অপারেন্ডই রেজিস্টার — কোনো মেমোরি অ্যাক্সেস নেই। |

## 🧩 উদাহরণ

ধরা যাক:

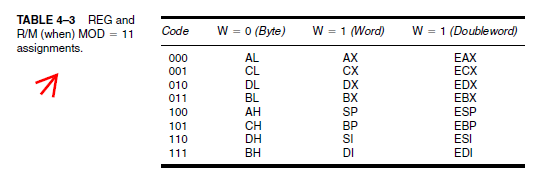
MOV AX, [BX+05H]

* এখানে BX হলো একটি বেস রেজিস্টার
* 05H হলো একটি ৮-বিট ডিসপ্লেসমেন্ট
* তাই **MOD = 01** (মেমোরি + ৮-বিট অফসেট)

ফলে, প্রসেসর বুঝবে যে **ডেটা মেমোরির (BX + 05H) ঠিকানায় আছে**।

## 🧠 সারসংক্ষেপ

| **Field** | **ভূমিকা** |
| --- | --- |
| **MOD** | ঠিকানা নির্ধারণ পদ্ধতি নির্দেশ করে (মেমোরি বা রেজিস্টার) |
| **REG** | অপারেশন-এ ব্যবহৃত রেজিস্টার নির্দেশ করে |
| **R/M** | মেমোরি বা রেজিস্টার অপারেন্ড নির্দেশ করে |



## 🔹 প্রেক্ষাপট

আগে তুমি যেটা শিখেছিলে, সেখানে বলেছিলাম যে **MOD-REG-R/M** বাইটে তিনটি ফিল্ড থাকে:

| **ফিল্ড** | **বিট সংখ্যা** | **কাজ** |
| --- | --- | --- |
| **MOD** | 2 bits | ঠিকানা নির্ধারণ পদ্ধতি (মেমোরি/রেজিস্টার) |
| **REG** | 3 bits | কোন রেজিস্টার ব্যবহার হবে |
| **R/M** | 3 bits | দ্বিতীয় রেজিস্টার বা মেমোরি লোকেশন নির্দেশ করে |

যখন **MOD = 11**, তখন প্রসেসর বুঝে যে —  
👉 উভয় অপারেন্ডই **রেজিস্টার**, কোনো মেমোরি অ্যাক্সেস নেই।

এই অবস্থায় **REG** এবং **R/M** ফিল্ডের মান অনুযায়ী রেজিস্টার বেছে নেওয়া হয়।

## 🔸 TABLE 4-3 ব্যাখ্যা

| **Code (REG বা R/M ফিল্ড)** | **W = 0 (Byte)** | **W = 1 (Word)** | **W = 1 (Doubleword)** |
| --- | --- | --- | --- |
| 000 | AL | AX | EAX |
| 001 | CL | CX | ECX |
| 010 | DL | DX | EDX |
| 011 | BL | BX | EBX |
| 100 | AH | SP | ESP |
| 101 | CH | BP | EBP |
| 110 | DH | SI | ESI |
| 111 | BH | DI | EDI |

### 🔍 ব্যাখ্যা:

* **Code:** REG বা R/M ফিল্ডের ৩-বিট মান (binary code)।
* **W bit:** অপারেন্ডের আকার নির্ধারণ করে।
  + W = 0 → 8-bit রেজিস্টার (byte)
  + W = 1 → 16-bit রেজিস্টার (word)
  + W = 1 (Doubleword) → 32-bit রেজিস্টার (Pentium class প্রসেসর থেকে)

### 🧩 উদাহরণ:

যদি নির্দেশনা হয়:

MOV AX, BX

তাহলে:

* MOD = 11 (রেজিস্টার-টু-রেজিস্টার অপারেশন)
* REG (AX) = 000
* R/M (BX) = 011

অর্থাৎ, এটি 16-bit ওয়ার্ড ডেটা স্থানান্তর করবে।

### 🧠 সারসংক্ষেপ:

* **MOD = 11 → উভয় অপারেন্ডই রেজিস্টার।**
* **REG ও R/M ফিল্ডের মান** নির্ধারণ করে কোন রেজিস্টার ব্যবহার হবে।
* **W bit** ঠিক করে রেজিস্টারের আকার — 8, 16, বা 32-bit।

চাও কি আমি তোমার জন্য একটি **ডায়াগ্রামসহ MOD-REG-R/M বাইটের পুরো গঠন** (বিট-লেভেল ভিজ্যুয়াল সহ) তৈরি করে দিই, যেখানে MOD = 11 দেখানো থাকবে?