ঠিক আছে 🙂 আমি তোমাকে এই অংশের মূল ধারণা সহজভাবে এবং ধাপে ধাপে বোঝাচ্ছি। আমরা দুইটি প্রধান বিষয় দেখব: **ENIAC** এবং **Von Neumann মেশিন**।

**১. ENIAC (Electronic Numerical Integrator And Computer)**

**পটভূমি:**

* ২য় বিশ্বযুদ্ধের সময়, U.S. Army **Ballistics Research Laboratory (BRL)** নতুন অস্ত্রের জন্য **range ও trajectory tables** তৈরি করতে সমস্যায় ছিল।
* এই হিসাবগুলো খুব সময় সাপেক্ষ ছিল এবং এক জন মানুষ একে একে হিসাব করলে অনেক ঘন্টা বা দিন লাগতো।

**কীভাবে সমাধান করা হলো:**

* **John Mauchly** (প্রফেসর) এবং **John Eckert** (ছাত্র) প্রস্তাব দিলেন একটি **general-purpose electronic computer** বানানোর।
* **1943**-এ Army অনুমোদন দিলো এবং ENIAC-এর কাজ শুরু হলো।

**ENIAC-এর বৈশিষ্ট্য:**

* ওজন: 30 tons
* জায়গা: 1500 sq. ft
* ১৮,০০০+ vacuum tubes
* শক্তি: 140 kW
* ৫,০০০ addition per second (অত্যন্ত দ্রুত)
* **Decimal machine** (binary নয়)
* **Memory:** 20 accumulators, প্রতিটা 10-digit ধরে রাখতে পারে।
* **Drawback:** প্রোগ্রাম করতে manually switches ও wires change করতে হতো।

**ব্যবহার:**

* 1946-এ ENIAC সম্পূর্ণ হলো, তাই WWII তে ব্যবহার হয়নি।
* প্রথম কাজে hydrogen bomb feasibility calculation এ ব্যবহার হয়েছিল।
* 1955 পর্যন্ত BRL পরিচালনায় চলেছিল।

**মূল কথা:** ENIAC হলো **প্রথম general-purpose electronic computer**, কিন্তু এটি **program manually** করতে হতো।

**২. Von Neumann মেশিন (Stored-program concept)**

**সমস্যা:**

* ENIAC-এ প্রোগ্রাম manually set করা খুব tedious ছিল।

**সমাধান:**

* **Stored-program concept:** প্রোগ্রামও **মেমরিতে ডেটার মতো স্টোর** করা যায়। তখন কম্পিউটার **মেমরি থেকে instructions পড়বে**।
* ধারণাটি **John von Neumann** এবং কিছুটা **Alan Turing** দিয়েছেন।
* প্রথম প্রস্তাবনা 1945 সালে EDVAC-এর জন্য হয়েছে।

**IAS Computer Structure (Prototype of all modern computers):**

1. **Main Memory:** Data + Instructions একসাথে সংরক্ষণ করে।
2. **Arithmetic & Logic Unit (ALU):** Binary data নিয়ে operations করে।
3. **Control Unit:** Instructions interpret করে এবং execute করে।
4. **Input/Output (I/O):** Control Unit দ্বারা পরিচালিত।

**Von Neumann-এর মূল বর্ণনা:**

* Arithmetic operations (add, subtract, multiply, divide) আলাদা organ দ্বারা করা হবে।
* Logical control বা operation sequencing করবে **central control organ (CC)**।
* Program instructions মেমরিতে থাকবে, এবং control unit তা execute করবে।

**মূল কথা:**

* Modern computer এর সবকটি element **Von Neumann architecture** থেকে এসেছে।
* Stored-program concept মেশিনকে **flexible ও general-purpose** বানায়।

**সংক্ষেপে মূল বিষয়গুলো**

| **বিষয়** | **ENIAC** | **Von Neumann Machine** |
| --- | --- | --- |
| সময় | 1943–46 | 1946–52 (IAS) |
| উদ্দেশ্য | Specific task (ballistics) | General-purpose |
| Program | Manual switches/cables | Stored in memory |
| Memory | Decimal accumulators | Binary memory (data + instructions) |
| Speed | 5000 additions/sec | আরও দ্রুত ও flexible |
| গুরুত্ব | প্রথম electronic computer | সব modern computer-এর prototype |

ঠিক আছে 🙂 আমি এই অংশটিকে ধাপে ধাপে বাংলায় সহজভাবে বুঝিয়ে দিচ্ছি। এখানে **Von Neumann-এর পাঁচটি মূল অংশ (specific parts)** বর্ণনা করা হয়েছে।

**Von Neumann-এর পাঁচটি specific parts**

**১️⃣ Central Arithmetic (CA)**

* কম্পিউটারকে **মূল arithmetic operations** করতে হবে, যেমন:
  + **যোগ (Addition)**
  + **বিয়োগ (Subtraction)**
  + **গুণ (Multiplication)**
  + **ভাগ (Division)**
* এজন্য একটি **specialized organ** দরকার, যা শুধু এই কাজগুলো করবে।
* এটিকে বলা হলো **CA (Central Arithmetic part)**।
* সহজভাবে: **মস্তিষ্কের “গাণিতিক অংশ”**।

**২️⃣ Central Control (CC)**

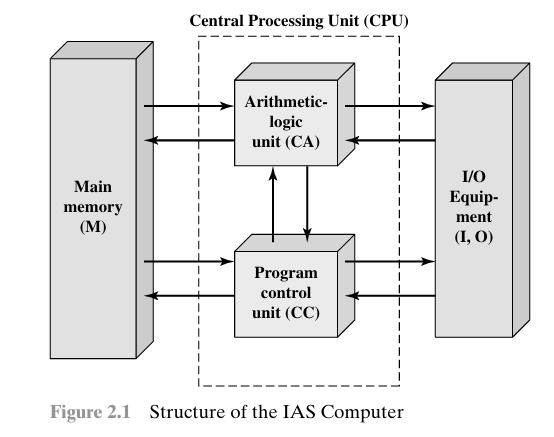
* কম্পিউটারের operations ঠিকভাবে **ক্রম অনুযায়ী চালাতে** হবে।
* এটি সম্পন্ন করে **central control organ (CC)**।
* এখানে দুইটা জিনিস আলাদা করা হয়:
  1. **Specific instructions:** নির্দিষ্ট সমস্যার জন্য।
  2. **General control organs:** instructions execute করবে যেভাবেই হোক।
* সহজভাবে: **মস্তিষ্কের “নিয়ন্ত্রণ বা sequencing অংশ”**, যা সব কাজের নির্দেশ দেয়।

## ৩️ Memory (M)

* যেকোনো কম্পিউটার যদি **দীর্ঘ ও জটিল calculations** করতে চায়, তাহলে তার **considerable memory (প্রচুর মেমরি)** থাকতে হবে।
* **Instructions** ও **programs**, যা জটিল সমস্যা সমাধান করে, তা অনেক জায়গা নেয়।
* এই **মোট memory** হলো কম্পিউটারের **তৃতীয় specific part: M**।

**সহজভাবে:**

* M হলো **কম্পিউটারের স্মৃতি**, যেখানে **ডেটা + প্রোগ্রাম instructions** রাখা হয়।
* এটি মানুষের মস্তিষ্কের **associative neurons** এর সমতুল্য।



**Input (I) এবং Output (O)**

* CA + CC + M হলো **কম্পিউটারের processing part (C + M)**।
* মানুষের মতো, কম্পিউটারেরও লাগবে **sensor এবং motor equivalents**:
  + **Input (I):** বাহ্যিক তথ্য নেয় → sensory neurons সমতুল্য
  + **Output (O):** প্রক্রিয়াজাত ফলাফল দেয় → motor neurons সমতুল্য
* এই বাহ্যিক তথ্য এবং ফলাফল রাখার জন্য একটা **outside recording medium (R)** দরকার।
  + Input থেকে R → M তে যাবে।
  + M থেকে R → Output হবে।

**সহজভাবে:**

* R = কম্পিউটারের “বাহ্যিক ডেটা storage” বা record medium
* I = Input organ (sensor)
* O = Output organ (motor)

🔑 **মূল ধারণা:**

1. **M** → ডেটা ও প্রোগ্রাম সংরক্ষণ
2. **C (CA + CC)** → প্রক্রিয়াজাত / হিসাব
3. **I/O** → বাহ্যিক জগতের সাথে যোগাযোগ
4. **R** → তথ্যের বাহ্যিক medium

ঠিক আছে 🙂 আমি এই অংশটিকে সহজভাবে বাংলায় বোঝাচ্ছি। এখানে **Input ও Output-এর কাজ এবং তথ্যের প্রবাহ** বোঝানো হয়েছে।

**৪️⃣ Input (I)**

* কম্পিউটারকে বাহ্যিক **recording medium (R)** থেকে তথ্য নিতে হবে।
* এই কাজটি করে **Input organ (I)**।
* **Important point:**
  + তথ্য **R → M** তে যাবে।
  + সরাসরি **R → C** এ না যাওয়াই ভালো।
* **সহজভাবে:**
  + Input organ হলো **sensor** বা **data reader**, যা সব তথ্য প্রথমে **memory (M)**-তে রাখে।

**৫️⃣ Output (O)**

* প্রক্রিয়াজাত তথ্য/ফলাফল **C + M থেকে বাহ্যিক medium (R)** পাঠাতে হবে।
* এই কাজটি করে **Output organ (O)**।
* **Important point:**
  + সব transfers **M → R** এর মাধ্যমে হবে।
  + **C → R** সরাসরি না।
* **সহজভাবে:**
  + Output organ হলো **display, printer বা actuator**, যা **memory থেকে তথ্য নিয়ে বাইরের জগতে পাঠায়**।

**তথ্য প্রবাহের মূল নিয়ম**

1. Input: **R → I → M → C**
2. Processing: **C (CA + CC)** + **M**
3. Output: **C + M → O → R**

**চিত্রের ভাবনা:**

R (External medium)

↑ ↓

O I

↑ ↓

M ←→ C (CA + CC)

🔑 **মূল ধারণা:**

* সব **input প্রথমে memory (M)** তে যায়।
* সব **output memory (M)** থেকে বের হয়।
* CA + CC সরাসরি input বা output-এর সাথে যুক্ত নয়।

**সংক্ষেপে Five Specific Parts**

| **Part** | **কাজ** | **Human Equivalent** |
| --- | --- | --- |
| **CA** | Arithmetic operations | Associative neurons (গাণিতিক অংশ) |
| **CC** | Control / sequencing | Associative neurons (নিয়ন্ত্রণ অংশ) |
| **M** | Memory (data + instructions) | Associative neurons (স্মৃতি) |
| **I** | Input (external data) | Sensory neurons |
| **O** | Output (results) | Motor neurons |

🔑 **মন্তব্য:**

* CA + CC একত্রে **C** নামেও পরিচিত।
* সব Input/Output মেমরি M-এর মাধ্যমে হয়, সরাসরি CA থেকে নয়।

ঠিক আছে 🙂 আমি তোমাকে এই অংশটি সহজভাবে বাংলায় বোঝাচ্ছি। এখানে মূলত **IAS কম্পিউটারের মেমরি এবং instruction format** নিয়ে আলোচনা করা হয়েছে।

**IAS Computer – মেমরি ও instruction structure**

**১️⃣ Memory Structure**

* IAS কম্পিউটারের **মেমরি** এ **1000 storage locations** আছে।
* প্রতিটি location = **1 word**, যা **40 bits** এর সমান।
* **Data + Instructions** একই memory-তে রাখা হয় (stored-program concept)।

**২️⃣ Number Representation**

* প্রতিটি number **binary form** এ থাকে।
* প্রতিটি number-এর:
  + **Sign bit** → number-এর positive/negative নির্দেশ করে
  + **39-bit value** → number-এর magnitude

**সহজভাবে:**

* Binary system ব্যবহার করা হয় কারণ এটি electronic computation-এর জন্য উপযুক্ত।

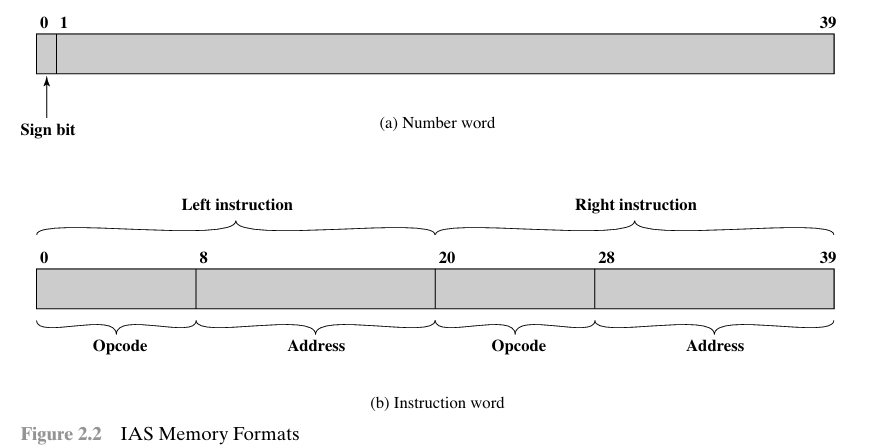
**৩️⃣ Instruction Representation**

* প্রতিটি word-এ **2টি 20-bit instruction** থাকতে পারে।
* প্রতিটি instruction:
  1. **Opcode (8-bit)** → কোন operation করা হবে তা নির্ধারণ করে (যেমন add, subtract)।
  2. **Address (12-bit)** → কোন memory location (0–999) ব্যবহার হবে তা নির্দেশ করে।

**সহজভাবে:**

* Opcode = কম্পিউটারকে বলে কি করতে হবে
* Address = কম্পিউটারকে বলে কোথায় data আছে বা কোথায় save করতে হবে

**উদাহরণভাবে**

**মোট কথা:**

* IAS কম্পিউটার **Von Neumann architecture** অনুসরণ করে
* Memory = data + program storage
* Binary number system ব্যবহার করে
* Instructions binary code আকারে store হয়
* প্রতিটি word-এ 2টি instruction থাকতে পারে

যদি চাও, আমি এটা **সরাসরি IAS computer word format diagram** বানিয়ে দেখাতে পারি, যা এক নজরে সব বোঝা যাবে।

চাও আমি সেটা করি?

ঠিক আছে 🙂 আমি এই অংশটি সহজভাবে বাংলায় ব্যাখ্যা করি। এখানে IAS কম্পিউটারের **registers** এবং তাদের কাজ বোঝানো হয়েছে।

**IAS Computer-এর Registers এবং তাদের কাজ**

**১️⃣ Memory Buffer Register (MBR)**

* MBR হলো একটি **temporary storage**।
* কাজ:
  + Memory থেকে data নিয়ে আসে বা memory-তে পাঠায়।
  + I/O unit-এর সাথে data transfer করতে ব্যবহার হয়।
* সহজভাবে: **MBR = memory এবং I/O-এর মধ্যে data carrier**।

**২️⃣ Memory Address Register (MAR)**

* MAR নির্দেশ করে **মেমরির কোন ঠিকানায় (address) data write বা read হবে**।
* MBR + MAR একসাথে memory access করে।
* সহজভাবে: **MAR = memory-এর ঠিকানা ধরার register**।

**৩️⃣ Instruction Register (IR)**

* IR ধারণ করে **current instruction-এর 8-bit opcode** যা ALU বা control unit execute করছে।
* সহজভাবে: **IR = current instruction execute করার জন্য register**।

**৪️⃣ Instruction Buffer Register (IBR)**

* প্রতিটি memory word-এ **2টি instruction থাকতে পারে**।
* IBR সাময়িকভাবে ধরে রাখে **word-এর right-hand instruction**।
* সহজভাবে: **IBR = next instruction temporarily store করার জন্য**।

**৫️⃣ Program Counter (PC)**

* PC ধারণ করে **পরবর্তী instruction-pair-এর memory address**, যা fetch করা হবে।
* সহজভাবে: **PC = computer-এর “pointer” যা বলে পরবর্তী কোন instruction execute হবে**।

**৬️⃣ Accumulator (AC) এবং Multiplier Quotient (MQ)**

* ALU operation-এর **operands এবং results temporarily store** করার জন্য।
* উদাহরণ: দুটি 40-bit number multiply করলে ফলাফল 80-bit হয়।
  + Most significant 40-bit → AC
  + Least significant 40-bit → MQ
* সহজভাবে: **AC + MQ = ALU-এর temporary storage**।

**সারসংক্ষেপ – Register Roles**

| **Register** | **কাজ** |
| --- | --- |
| **MBR** | Memory ↔ I/O-এর মধ্যে data transfer |
| **MAR** | Memory-এর address নির্ধারণ |
| **IR** | Current instruction execute করা |
| **IBR** | Right-hand instruction temporarily রাখা |
| **PC** | Next instruction-এর address ধরে রাখা |
| **AC + MQ** | ALU operation-এর operands ও results ধরে রাখা |

যদি চাও, আমি IAS computer-এর **সকল registers এবং memory এর সংযোগসহ একটি diagram** বানিয়ে দেখাতে পারি, যা এক নজরে সব relationship বোঝাবে।

চাও আমি সেটা করি?

খুব ভালো প্রশ্ন 🙂  
এখানে IAS কম্পিউটারের **Instruction Cycle** এবং এর ধাপগুলো ব্যাখ্যা করা হয়েছে। আমি বাংলায় সহজভাবে ধাপে ধাপে বুঝিয়ে দিচ্ছি।

**IAS Instruction Cycle**

IAS কম্পিউটার সবসময় **একটি লুপে কাজ করে** → Instruction Cycle।  
এটি আবার দুই ভাগে বিভক্ত:

**১️⃣ Fetch Cycle**

* কাজ: **পরবর্তী instruction মেমরি থেকে আনা**।
* ধাপ:
  1. **PC (Program Counter)** → কোন instruction আনতে হবে তার address রাখে।
  2. সেই address MAR (Memory Address Register)-এ যায়।
  3. ওই address থেকে word memory থেকে MBR (Memory Buffer Register)-এ আসে।
  4. যেহেতু ১ word = ২ instruction (20-bit + 20-bit),
     + Left-hand instruction সরাসরি IR + MAR এ যায়।
     + Right-hand instruction IBR (Instruction Buffer Register)-এ সাময়িকভাবে রাখা হয়।
* **কারণ (Indirection):** সব memory access-এর জন্য শুধু **একটা MAR** আর data transfer-এর জন্য **একটা MBR** ব্যবহার হয়। এতে ইলেকট্রনিক সার্কিট design সহজ হয়।

**২️⃣ Execute Cycle**

* কাজ: **Instruction execute করা**।
* ধাপ:
  1. IR-এ থাকা opcode control circuitry পড়ে।
  2. Control Unit → ALU, Memory, I/O এর মধ্যে প্রয়োজনীয় **control signals** পাঠায়।
  3. Instruction অনুযায়ী data transfer, ALU computation, বা branch operation হয়।

IAS-এর মোট **২১টি instruction** ছিল, আর সেগুলোকে **৫টি ক্যাটেগরিতে** ভাগ করা হয়। প্রতিটা ক্যাটেগরি আমি ধাপে ধাপে সহজভাবে বোঝাই 👇

**🔹 ১. Data Transfer Instruction**

👉 কাজ: ডেটা এক জায়গা থেকে আরেক জায়গায় নেওয়া/আনা।

* Memory ↔ ALU register
* Register ↔ Register

📌 উদাহরণ:

* **LOAD (Load AC from memory)** → মেমোরি থেকে ডেটা এনে Accumulator (AC)-এ রাখা।
* **STORE (Store AC to memory)** → AC থেকে ডেটা মেমোরিতে রাখা।

💡 ব্যবহার: ধরো, তুমি মেমোরি থেকে একটা সংখ্যা নিয়ে আসতে চাও যাতে সেটা দিয়ে হিসাব করা যায় → LOAD ব্যবহার করবে।

**🔹 ২. Unconditional Branch Instruction**

👉 কাজ: **প্রোগ্রামের flow পরিবর্তন করা**, সরাসরি অন্য address-এ লাফ দেওয়া।

* Program Counter (PC)-এর মান পাল্টে যায় → ফলে instruction execution অন্য address থেকে শুরু হয়।

📌 উদাহরণ:

* **JUMP** → সবসময় নির্দিষ্ট address-এ চলে যাবে।

💡 ব্যবহার: লুপ বানাতে বা একই instruction বারবার চালাতে। যেমন: goto স্টাইল।

**🔹 ৩. Conditional Branch Instruction**

👉 কাজ: শর্ত দিয়ে প্রোগ্রামের flow পরিবর্তন করা।

* যদি শর্ত **TRUE** হয় → PC বদলাবে।
* যদি **FALSE** হয় → প্রোগ্রাম normal sequence-এ চলবে।

📌 উদাহরণ:

* **JUMP + condition (e.g., JUMP if AC = 0)** → Accumulator শূন্য হলে লাফ দেবে, নাহলে না।

💡 ব্যবহার: প্রোগ্রামে decision নেওয়ার জন্য, যেমন → *if-else structure*।

**🔹 ৪. Arithmetic Instruction**

👉 কাজ: গাণিতিক হিসাব করা।

* যোগ, বিয়োগ, গুণ, ভাগ ইত্যাদি সব ALU করে।

📌 উদাহরণ:

* **ADD X** → AC = AC + M(X)
* **SUB X** → AC = AC – M(X)
* **MUL X** → AC = AC × M(X)
* **DIV X** → AC = AC ÷ M(X)

💡 ব্যবহার: সরাসরি গণনা করার জন্য, যেমন দুইটা সংখ্যার যোগফল বের করা।

**🔹 ৫. Address Modify Instruction**

👉 কাজ: Address নিজেই পরিবর্তন করা।

* ALU দিয়ে নতুন address হিসাব করে instruction-এর address field-এ ঢুকিয়ে দেয়।

📌 উদাহরণ:

* **Modify Address Instruction** → মেমোরিতে থাকা instruction-এর operand address বদলে ফেলে।

💡 ব্যবহার:

* Array indexing (প্রতিবার নতুন element access করার জন্য address বাড়ানো)
* Dynamic data access (প্রোগ্রাম চলার সময় address পরিবর্তন)

**📝 সারসংক্ষেপ টেবিল:**

| **ক্যাটেগরি** | **কী করে** | **উদাহরণ** |
| --- | --- | --- |
| Data Transfer | ডেটা আনা-নেওয়া | LOAD, STORE |
| Unconditional Branch | সবসময় jump | JUMP |
| Conditional Branch | শর্তসাপেক্ষ jump | JUMP if AC=0 |
| Arithmetic | হিসাব-নিকাশ | ADD, SUB, MUL, DIV |
| Address Modify | Address পরিবর্তন | Modify instruction address |

**সংক্ষেপে Instruction Cycle Flow**

FETCH:

PC → MAR → Memory → MBR

↓

IR ← Left Instruction

IBR ← Right Instruction

EXECUTE:

IR (opcode) → Control Unit → ALU/Memory/I-O

👉 এক কথায়: IAS কম্পিউটার প্রতিবার **Fetch → Execute** সাইকেল করে কাজ করে।  
প্রথমে instruction memory থেকে আনে, তারপর তা execute করে।  
Instruction set ছোট হলেও (২১টা), এর মধ্যেই data transfer, control flow (branch), arithmetic, এবং address modification সবকিছু ছিল।

দারুণ 🙂 তুমি যে অংশটা পাঠিয়েছো সেটা মূলত **১৯৫০ দশকে কমার্শিয়াল কম্পিউটারের ইতিহাস** ব্যাখ্যা করছে। আমি সহজভাবে ব্যাখ্যা করছি —

**🌍 ১৯৫০-এর দশকে কম্পিউটার ইন্ডাস্ট্রির সূচনা**

* ১৯৫০-এর দশককে বলা হয় কম্পিউটার ইন্ডাস্ট্রির জন্মকাল।
* এই সময়ের প্রধান দুটি কোম্পানি ছিল:
  1. **Sperry (UNIVAC)**
  2. **IBM (International Business Machines)**

**🖥️ UNIVAC (Sperry Rand)**

* **১৯৪৭**: Eckert এবং Mauchly → *Eckert-Mauchly Computer Corporation* প্রতিষ্ঠা করেন।
* তাদের প্রথম সফল কম্পিউটার: **UNIVAC I (Universal Automatic Computer)**।
* **১৯৫০ সালে U.S. Census Bureau** এটাকে ব্যবহার করে গণনার জন্য।
* পরে কোম্পানিটি → **Sperry-Rand Corporation**-এর UNIVAC division হয়ে যায়।

📌 **UNIVAC I এর গুরুত্ব**:

* প্রথম সফল **commercial computer**।
* ব্যবহার হয়েছিল → বৈজ্ঞানিক হিসাব, পরিসংখ্যান, বীমা কোম্পানির প্রিমিয়াম বিলিং, লজিস্টিক সমস্যা সমাধান ইত্যাদিতে।

📌 **UNIVAC II**:

* UNIVAC I-এর তুলনায় বেশি memory আর পারফরম্যান্স।
* একটি নতুন ট্রেন্ড শুরু করল →
  1. **প্রযুক্তির অগ্রগতির সাথে সাথে আরও শক্তিশালী কম্পিউটার তৈরি।**
  2. **Backward compatibility**: নতুন মেশিনে পুরনো প্রোগ্রাম চালানো যাবে → এতে পুরনো গ্রাহকরা হারাবে না।

📌 **UNIVAC 1100 series**:

* UNIVAC-এর সবচেয়ে গুরুত্বপূর্ণ সিরিজ।
* যেমন **UNIVAC 1103** → মূলত বৈজ্ঞানিক কাজে (লম্বা, জটিল হিসাব) ব্যবহৃত হত।
* তখনকার বাজারে একটা স্পষ্ট **scientific vs business computing** বিভাজন ছিল।

**🖥️ IBM (International Business Machines)**

* IBM মূলত **punched-card processing machine**-এর জন্য বিখ্যাত ছিল।
* **১৯৫৩ সালে IBM 701** প্রকাশ করে → এটা ছিল প্রথম stored-program electronic computer।
  + মূলত **scientific applications**-এর জন্য।
* **১৯৫5 সালে IBM 702** প্রকাশ করে → এতে হার্ডওয়্যার business applications-এর জন্য উপযুক্ত ছিল।
* এগুলো থেকে IBM-এর **700/7000 series** তৈরি হয় → যা IBM-কে বিশ্বের সবচেয়ে বড় কম্পিউটার নির্মাতা করে তোলে।

**📝 সারসংক্ষেপ (সহজ ভাষায়)**

* **UNIVAC I** → প্রথম commercial computer (scientific + business কাজ)।
* **UNIVAC II** → বড় memory, বেশি গতি, backward compatibility।
* **UNIVAC 1100 series** → বৈজ্ঞানিক গবেষণায় ব্যবহৃত।
* **IBM 701 (1953)** → বৈজ্ঞানিক কাজে।
* **IBM 702 (1955)** → ব্যবসায়িক কাজে।
* 1950s-এ → Sperry (UNIVAC) আর IBM ছিল মূল প্রতিযোগী।

👉 চাইলে আমি তোমাকে একটা **timeline চার্ট** বানিয়ে দিতে পারি যেখানে UNIVAC আর IBM-এর 1950s এর মেশিনগুলো এক নজরে দেখা যাবে।  
তুমি কি সেটা চাও?

এটা মূলত **Second Generation Computers (Transistor যুগ, 1955–1964)** নিয়ে লেখা। আমি সহজ ভাষায় ভেঙে বুঝিয়ে দিচ্ছি—

**🔹 Vacuum Tube → Transistor পরিবর্তন**

* প্রথম প্রজন্মে (1940s–early 1950s) কম্পিউটারে **Vacuum tube** ব্যবহার হতো।
* সমস্যা: বড় আকার, বেশি গরম হওয়া, দামি, প্রচুর বিদ্যুৎ খরচ।
* সমাধান: **Transistor (1947, Bell Labs)** → ছোট, সস্তা, কম গরম হয়, solid-state (সিলিকন দিয়ে বানানো)।
* 1950s-এর শেষে বাজারে **transistorized computer** পাওয়া শুরু হয়।

👉 এই পরিবর্তনেই শুরু হয় **Second Generation of Computers**।

**🔹 Second Generation-এর বৈশিষ্ট্য**

1. **Transistor ব্যবহার** → Vacuum tube-এর জায়গায়।
2. **আকার ছোট** + **দ্রুতগতি** + **কম বিদ্যুৎ খরচ**।
3. **বেশি Memory Capacity** (KiloWords থেকে কয়েক দশ হাজার word পর্যন্ত)।
4. **High-Level Programming Language** → FORTRAN, COBOL ইত্যাদি চালু হয়।
5. **System Software** যুক্ত হয় (Assembler, Compiler)।
6. **Control Unit & ALU উন্নত হয়** (আরও জটিল operation করতে পারে)।

**🔹 নতুন কোম্পানি: DEC (Digital Equipment Corporation)**

* প্রতিষ্ঠিত হয় **1957 সালে**।
* প্রথম কম্পিউটার: **PDP-1 (Programmed Data Processor-1)**।
* এটা ছিল ছোট ও সস্তা → **minicomputer trend** শুরু করে।
* পরবর্তী প্রজন্মে (third generation) DEC বাজারে বড় ভূমিকা রাখে।

**🔹 IBM 7000 সিরিজ**

* IBM vacuum tube যুগে (700 series, 1952 থেকে শুরু) → পরবর্তীতে transistor ভিত্তিক **7000 series** বের করে।
* এর মধ্যে সবচেয়ে গুরুত্বপূর্ণ → **IBM 7094** (early 1960s)।

📌 উন্নতি:

* **Memory size**: 2K → 32K words (36-bit word)।
* **Memory access time**: 30 µs → 1.4 µs।
* **Instruction (Opcode)** সংখ্যা: 24 → 185।
* **Execution Speed** অনেক বেড়ে যায় (transistor + জটিল সার্কিট)।

**🔹 IBM 7094-এর নতুন বৈশিষ্ট্য**

1. **Instruction Prefetching** →
   * একসাথে দুটি instruction মেমোরি থেকে আনা হয়।
   * ফলে প্রতি cycle-এ নতুন instruction আনতে হয় না → গতি বেড়ে যায়।
2. **Data Channel (I/O Processor)** →
   * আলাদা প্রসেসর ছিল I/O কাজের জন্য।
   * CPU শুধু control signal পাঠাতো → I/O processor নিজে কাজ শেষ করে CPU-কে জানাতো।
   * এতে CPU-র উপর লোড অনেক কমে যায়।
3. **Multiplexor** →
   * CPU, Data Channel, Memory — সবাই মেমোরি শেয়ার করত।
   * Multiplexor schedule করে দিত কে কখন মেমোরি ব্যবহার করবে।
   * ফলে সব ডিভাইস স্বাধীনভাবে কাজ করতে পারত।

**📝 সারসংক্ষেপ**

* **Second Generation (1955–1964)** = Transistor যুগ।
* বৈশিষ্ট্য → ছোট, দ্রুত, নির্ভরযোগ্য, সস্তা, বেশি memory, high-level language, system software।
* IBM 7000 series (বিশেষত IBM 7094) → উন্নত speed, prefetching, data channel, multiplexor।
* DEC PDP-1 → mini-computer যুগের সূচনা।

👉 চাইলে আমি তোমাকে একটা **comparison table বানিয়ে দিতে পারি** (First vs Second Generation computers), যাতে পার্থক্য এক নজরে বোঝা যায়।  
তুমি কি সেটা চাইছো?

খুব সুন্দর অংশ পাঠিয়েছো 🙂  
এটা মূলত **Third Generation of Computers (Integrated Circuit যুগ, 1964–1971)** নিয়ে লেখা। আমি সহজ ভাষায় ব্যাখ্যা করছি —

**🔹 Third Generation শুরু: Integrated Circuits (IC)**

* **প্রথম ও দ্বিতীয় প্রজন্মে** → আলাদা আলাদা component (transistor, resistor, capacitor) নিয়ে সার্কিট বানাতে হতো।
* সমস্যা:
  + প্রচুর component → soldering জটিল, খরচ বেশি, ত্রুটি হওয়ার সম্ভাবনা বেশি।
  + Second gen-এ কম্পিউটারগুলোতে হাজার হাজার transistor ছিল → তৈরি ও maintain করা কঠিন হয়ে যাচ্ছিল।

👉 সমাধান: **1958 সালে Integrated Circuit (IC)** আবিষ্কার।

* অনেকগুলো transistor, resistor, conductor একসাথে ছোট একটা **silicon chip**-এ বসানো হলো।
* এর ফলে এক টুকরো silicon wafer-এ একসাথে অসংখ্য circuit বানানো সম্ভব হলো।
* এটাই **Third Generation of Computers**-এর মূল বৈশিষ্ট্য।

### ****Integrated Circuit (IC) এর সংজ্ঞা:****

**Integrated Circuit (IC)** হলো একটি ছোট **সেমিকন্ডাক্টর চিপ (সাধারণত সিলিকন)**, যেখানে **অনেকগুলো transistor, resistor, capacitor, এবং অন্যান্য electronic component** একসাথে যুক্ত থাকে এবং একটি নির্দিষ্ট কাজ করার মতো সার্কিট গঠন করে।

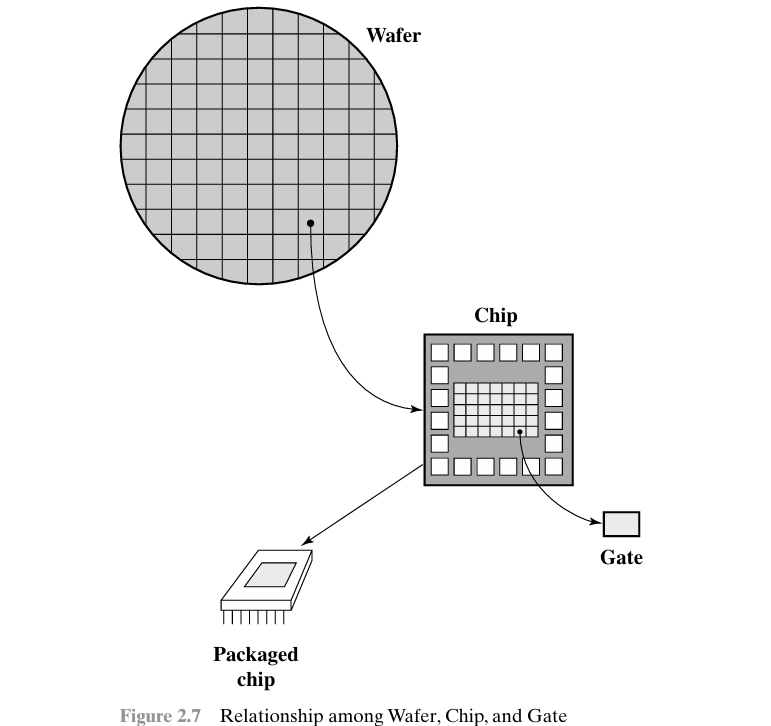
**🔹 Microelectronics**

* **Microelectronics = ছোট electronics** → আকার ছোট, কিন্তু ক্ষমতা বেশি।
* কম্পিউটারের মূল building block:
  1. **Gate** → লজিক্যাল কাজ করে (AND, OR ইত্যাদি)।
  2. **Memory cell** → 1 bit তথ্য ধরে রাখে।

👉 এই দুইটার সংযোগ দিয়েই পুরো কম্পিউটার তৈরি হয়।

**🔹 IC (Integrated Circuit)-এর উৎপাদন**

1. একটা **silicon wafer**-এ ছোট ছোট এলাকায় একই ধরনের সার্কিট তৈরি হয়।
2. প্রতিটা এলাকা ভেঙে আলাদা **chip** বানানো হয়।
3. প্রতিটা chip-এ থাকে → অনেকগুলো gate বা memory cell + input/output pins।
4. কয়েকটা chip মিলে একটা বড় সার্কিট তৈরি হয়।



**🔹 IC-এর প্রকারভেদ**

* **SSI (Small Scale Integration)** → প্রথম দিকে কয়েকটা transistor মাত্র।
* পরে transistor density বাড়তে থাকে → **MSI, LSI, VLSI** পর্যন্ত যায়।
* এ ট্রেন্ডটা Gordon Moore (Intel co-founder) **1965 সালে Moore’s Law** হিসেবে বলেছিলেন:
  + প্রতি বছর (পরে প্রতি ১৮ মাসে) chip-এ transistor সংখ্যা **দ্বিগুণ** হবে।
  + আজও এই ধারা বহাল আছে।

**🔹 Moore’s Law-এর প্রভাব**

1. **Cost কমেছে** → transistor সংখ্যা বাড়লেও chip-এর দাম প্রায় একই থাকছে।
2. **Speed বেড়েছে** → transistor একে অপরের কাছাকাছি থাকায় signal দ্রুত যায়।
3. **আকার ছোট হয়েছে** → কম্পিউটার ছোট ছোট ডিভাইসে বসানো সম্ভব।
4. **Power ও cooling প্রয়োজন কমেছে**।
5. **Reliability বেড়েছে** → solder connection কমে গেছে, circuit ভিতরেই সংযুক্ত।

**🖥️ Third Generation-এর গুরুত্বপূর্ণ উদাহরণ**

1. **IBM System/360 (1964)** →
   * প্রথমবার একটা computer family তৈরি হলো যেখানে ছোট-বড় সব মডেল একই instruction set ব্যবহার করে।
   * Commercial ও scientific কাজের জন্য একসাথে ব্যবহারযোগ্য।
2. **DEC PDP-8 (1965)** →
   * সস্তা, ছোট → **minicomputer যুগের** সূচনা।
   * শিক্ষাপ্রতিষ্ঠান, ছোট ব্যবসা → প্রথমবার কম্পিউটার ব্যবহার শুরু করল।

**📝 সারসংক্ষেপ**

* **Third Generation (1964–1971)** = Integrated Circuit (IC)-এর যুগ।
* বৈশিষ্ট্য → ছোট আকার, দ্রুতগতি, কম খরচ, কম বিদ্যুৎ, নির্ভরযোগ্য, mass production।
* **Moore’s Law** transistor density বাড়ার গতি ব্যাখ্যা করে।
* মূল কম্পিউটার: IBM System/360, DEC PDP-8।

তুমি যে অংশটা পাঠিয়েছো, সেখানে আসলে **IBM System/360** আর **DEC PDP-8** – এই দুইটা কম্পিউটার সিস্টেমের ঐতিহাসিক গুরুত্ব আর বৈশিষ্ট্য আলোচনা করা হয়েছে। আমি তোমাকে পরীক্ষার মতো **সহজ ভাষায় পয়েন্ট আকারে** সাজিয়ে দিচ্ছি 👇

**IBM System/360 (1964)**

🔑 **মূল পয়েন্টসমূহ:**

* IBM আগেই **7000 সিরিজ** দিয়ে মার্কেট দখল করেছিল।
* 1964 সালে IBM **System/360** ঘোষণা করে → যা পুরানো মেশিনের সাথে *incompatible* ছিল।
* কিন্তু এটা ছিল এক সাহসী পদক্ষেপ, কারণ 360 আর্কিটেকচার নতুন **IC (Integrated Circuit)** প্রযুক্তি ব্যবহার করেছিল।
* System/360 পুরো দশকের সেরা সাফল্য হয় → IBM-এর মার্কেট শেয়ার >70%।
* আজও কিছু পরিবর্তনসহ System/360 এর আর্কিটেকচার IBM মেইনফ্রেমে ব্যবহার হচ্ছে।

**মূল বৈশিষ্ট্য (Family Concept):**

1. **Similar Instruction Set** → এক মডেলের প্রোগ্রাম অন্য মডেলে রান করতে পারে।
2. **Same OS** → পুরো পরিবারের জন্য একই বেস অপারেটিং সিস্টেম।
3. **Increasing Speed** → ছোট মডেল ধীর, বড় মডেল দ্রুত।
4. **I/O Ports বেশি** → বড় মডেলে I/O সুবিধা বেশি।
5. **Memory Size বেশি** → ছোট থেকে বড় মডেলে মেমরি বাড়ে।
6. **Cost বেশি** → মডেল যত বড়, দাম তত বেশি।

👉 গ্রাহক চাইলে ছোট মডেল কিনে ভবিষ্যতে বড় মডেলে আপগ্রেড করতে পারত, সফটওয়্যার নষ্ট না করেই।  
👉 **Impact:** System/360 শুধু IBM না, পুরো কম্পিউটার ইন্ডাস্ট্রির দিক পাল্টে দেয়।

**DEC PDP-8 (1964)**

🔑 **মূল পয়েন্টসমূহ:**

* একই বছরে **DEC (Digital Equipment Corporation)** তৈরি করে **PDP-8**।
* এটাকে বলা হয়েছিল **minicomputer** → কারণ এটা ছোট (ল্যাব বেঞ্চে রাখা যায়)।
* দাম খুব কম → মাত্র **$16,000** (System/360 এর তুলনায় কয়েক লক্ষ ডলার সস্তা)।
* তাই প্রতিটি ল্যাব টেকনিশিয়ানের জন্যও সম্ভব ছিল।
* DEC PDP-8 মোট **৫০,০০০ মেশিন** বিক্রি হয়েছিল → বিশাল সাফল্য।

**মূল বৈশিষ্ট্য:**

1. ছোট আকার, কম দামে → জনপ্রিয়তা বিস্ফোরকভাবে বাড়ে।
2. **OEM Market** তৈরি করে → অনেক কোম্পানি PDP-8 কিনে নিজের সিস্টেম বানিয়ে বিক্রি করত।
3. DEC হয়ে ওঠে **নম্বর ১ Minicomputer Vendor** এবং **IBM-এর পর দ্বিতীয় বৃহত্তম কম্পিউটার কোম্পানি**।
4. PDP-8 এর পরের মডেলগুলোতে ব্যবহৃত হয়েছিল **Bus Architecture (Omnibus)** → যা আজকের মাইক্রোকম্পিউটারে standard।

**সংক্ষেপে তুলনা:**

| **বিষয়** | **IBM System/360** | **DEC PDP-8** |
| --- | --- | --- |
| টাইপ | Mainframe | Minicomputer |
| দাম | অনেক বেশি (লক্ষ ডলার) | অনেক কম ($16,000) |
| আকার | বড়, রুমে রাখতে হয় | ছোট, টেবিলে রাখা যায় |
| টার্গেট মার্কেট | বড় প্রতিষ্ঠান | ল্যাব, ছোট প্রতিষ্ঠান |
| বৈশিষ্ট্য | Family concept, compatibility | সস্তা, flexible, bus structure |
| প্রভাব | IBM mainframe dominance | Minicomputer industry শুরু |

খুব সুন্দর অংশটা তুমি শেয়ার করেছো 🙂 এখানে মূলত **তৃতীয় প্রজন্মের পরবর্তী কম্পিউটার জেনারেশন** গুলো কিভাবে ডিফাইন করা হয়, তারপরে **সেমিকন্ডাক্টর মেমরি** আর **মাইক্রোপ্রসেসরের জন্ম ও বিবর্তন** ব্যাখ্যা করা হয়েছে। আমি একে একে সংক্ষেপে সাজিয়ে দিচ্ছি 👇

**Later Generations of Computers**

* **প্রথম তিন প্রজন্ম** (Vacuum tube → Transistor → IC) স্পষ্টভাবে আলাদা করা যায়।
* এর পর থেকে **LSI, VLSI, ULSI** প্রযুক্তি এসেছে → কিন্তু প্রজন্মভিত্তিক বিভাজন স্পষ্ট না।

**IC Technology অনুযায়ী উন্নতি:**

1. **LSI (Large Scale Integration):**  
   → 1000+ কম্পোনেন্ট একটি চিপে।
2. **VLSI (Very Large Scale Integration):**  
   → 10,000+ কম্পোনেন্ট একটি চিপে।
3. **ULSI (Ultra Large Scale Integration):**  
   → 1 মিলিয়নেরও বেশি কম্পোনেন্ট একটি চিপে।

**Semiconductor Memory**

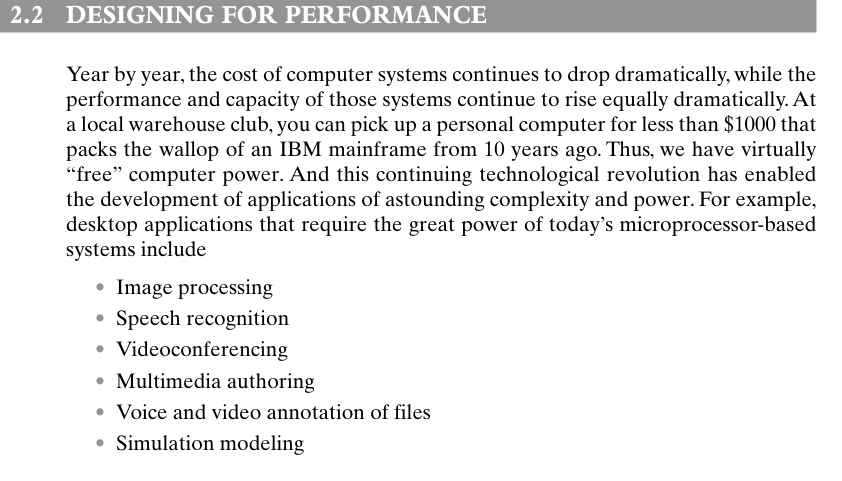
* **১৯৫০–৬০ দশক:** Memory তৈরি হতো **Magnetic Core** দিয়ে।
  + একেকটা ছোট রিং “1” বা “0” রিপ্রেজেন্ট করত।
  + দ্রুত (1 μs) কিন্তু ব্যয়বহুল ও bulky।
  + **Destructive readout** → পড়লেই ডেটা মুছে যেত, আবার restore করতে হতো।
* **1970 (Fairchild):** প্রথম উল্লেখযোগ্য **semiconductor memory chip** তৈরি হয় → 256 bits ধারণক্ষমতা।
  + nondestructive, দ্রুততর (70 ns), কিন্তু প্রথমে দাম বেশি।
* **1974:** **Semiconductor memory এর দাম core memory থেকে কমে যায়।**
  + এর পর থেকে দাম দ্রুত কমতে থাকে, আর capacity বাড়তে থাকে।
  + কম্পিউটার ছোট, সস্তা ও দ্রুত হয় → **personal computer era** শুরু হয়।
* **Memory generations:** 1K → 4K → 16K → 64K → 256K → 1M → 4M → 16M → 64M → 256M → 1G → 4G → 16G (এক চিপে)।
  + প্রতিবার **৪ গুণ বেশি storage** এবং কম খরচে পাওয়া গেছে।

**Microprocessors**

* **1971 (Intel 4004):**
  + প্রথম **single-chip CPU**।
  + 4-bit প্রসেসর, কেবল যোগ করতে পারে, গুণ করতে পারে শুধু repeated addition দ্বারা।
  + primitive হলেও **microprocessor যুগের সূচনা।**
* **1972 (Intel 8008):**
  + প্রথম **8-bit processor**।
  + 4004 থেকে দ্বিগুণ complex।
* **1974 (Intel 8080):**
  + প্রথম **general-purpose microprocessor**।
  + দ্রুততর, বড় instruction set, বড় addressing ক্ষমতা।
  + মাইক্রোকম্পিউটার তৈরি সম্ভব করে।
* **Late 1970s:** 16-bit microprocessor আসে।
  + **Intel 8086 (1978):** অন্যতম জনপ্রিয় 16-bit processor।
* **Early 1980s:** 32-bit microprocessor আসে।
  + Bell Labs, HP তৈরি করে (1981)।
  + **Intel 80386 (1985):** অন্যতম সফল 32-bit processor।

**সংক্ষেপে Impact**

* **Semiconductor memory + Microprocessor** একসাথে কম্পিউটারকে ছোট, সস্তা ও শক্তিশালী করে।
* এর ফলেই **personal computer revolution** শুরু হয়।



খুব সুন্দর অংশটা তুমি এনেছো 🙂 এখানে মূল বিষয় হলো—  
**কম্পিউটার পারফরম্যান্স বাড়ানোর জন্য কীভাবে ডিজাইন করতে হয়** (Designing for Performance)।

আমি সহজ ভাষায় সাজিয়ে দিচ্ছি 👇

# **Designing for Performance**

### ****১. Microprocessor Speed****

* **Moore’s Law** অনুযায়ী প্রতি ১৮–২৪ মাসে chip-এ ট্রানজিস্টরের সংখ্যা দ্বিগুণ হচ্ছে → Processor দ্রুততর হচ্ছে।
* Memory capacity (DRAM) ও একই হারে বাড়ছে।
* Problem → শুধু processor দ্রুত হলে হবে না, তাকে ক্রমাগত instruction দিতে হবে।

#### **Speed বাড়ানোর টেকনিকসমূহ:**

1. **Branch Prediction**
   * Processor অনুমান করে কোন instruction পরবর্তী হবে।
   * Guess ঠিক হলে → আগে থেকে instruction এনে buffer-এ রাখে → Processor idle থাকে না।
   * Advanced processor একাধিক branch-ও predict করতে পারে।
2. **Data Flow Analysis**
   * Instruction গুলির মধ্যে dependency দেখে কোনটা আগে execute হবে তা ঠিক করে।
   * Original order না মেনে, dependency অনুযায়ী execute করলে delay কমে।
3. **Speculative Execution**
   * Branch prediction + data flow ব্যবহার করে future-এর instruction আগে থেকেই execute করে।
   * Result temporary register-এ রাখা হয়।
   * যদি প্রেডিকশন সঠিক হয় → execution already done!

### ****২. Performance Balance****

* **Problem:** Processor speed দ্রুত বাড়ছে, কিন্তু **Memory speed** ততটা বাড়ছে না।
* Memory ←→ Processor interface ধীর হলে processor idle হয়ে যায় (wait state)।
* তাই balance করতে হবে → Processor, Memory, I/O সব মিলিয়ে।

#### **Solution / Design Techniques**

1. **Wider DRAM & Bus**
   * একসাথে বেশি bit transfer করা।
   * যেমন 8-bit এর বদলে 64-bit bus।
2. **Efficient DRAM Interface**
   * DRAM chip-এর মধ্যেই cache/buffer রাখা → দ্রুত access।
3. **Cache Memory ব্যবহার**
   * Processor chip-এর ভেতরে (L1, L2 cache) + বাইরে (L3 cache)।
   * Cache hit হলে main memory থেকে ডেটা আনতে হয় না → speed বাড়ে।
4. **High-Speed Buses**
   * Processor ↔ Memory ↔ I/O এর জন্য একাধিক hierarchy buses ব্যবহার।
   * Bandwidth বাড়ে → Data congestion কমে।

### ****৩. I/O Devices****

* Modern I/O devices (disk, network, graphics, camera ইত্যাদি) → প্রচুর data generate করে।
* Processor handle করতে পারে, কিন্তু data movement (I/O ↔ Processor) bottleneck হয়।

#### **Solution**

* I/O caching ও buffering।
* High-speed interconnection buses (PCIe ইত্যাদি)।
* Multiprocessor configuration → I/O load ভাগ করে নেয়।

### ****৪. Key Idea: Balance****

* সব জায়গায় balance করতে হবে:
  + Processor speed
  + Memory speed & size
  + I/O throughput
  + Interconnection bandwidth
* কারণ, এগুলির development rate সমান নয়।
* New application + নতুন peripheral → সবসময় নতুন demand তৈরি করে।

👉 এক কথায়:  
কম্পিউটার ডিজাইন মানে শুধু processor fast করা নয়, বরং **পুরো সিস্টেমকে balanced করা** যেন processor, memory আর I/O একসাথে efficient ভাবে কাজ করে।

# **Improvements in Chip Organization and Architecture**

Processor performance বাড়ানোর জন্য মূলত তিনটি ধাপ আছে:

### ****১. Hardware Speed বাড়ানো****

* Logic gate ছোট করা → একসাথে বেশি gate pack করা যায় → signal propagation দ্রুত হয়।
* Clock rate বাড়ানো → operations দ্রুত execute হয়।

### ****২. Cache Size এবং Speed বাড়ানো****

* Processor ↔ Memory মধ্যে cache ব্যবহার।
* Processor chip-এর মধ্যে cache থাকলে access time অনেক কমে।

### ****৩. Processor Organization & Architecture পরিবর্তন****

* Parallelism ব্যবহার করে instruction execution দ্রুত করা।
* উদাহরণ: Pipelining, Superscalar।

### ****Challenges / Obstacles****

1. **Power Problem**
   * High-density + high-speed → heat বেশি হয়।
   * Power dissipation এখন বড় design concern।
2. **RC Delay**
   * Wire resistance (R) এবং capacitance (C) বেড়ে যায় → signal delay বাড়ে।
3. **Memory Latency**
   * Memory এখনো processor-এর চেয়ে ধীরে কাজ করে → bottleneck।

### ****Performance Enhancement Techniques (1980s–2000s)****

1. **Cache Increase**
   * Multiple levels of cache (L1, L2, L3)।
   * On-chip cache বাড়ানো → faster access।
   * উদাহরণ: Pentium → 10% on-chip cache, Pentium 4 → ~50% on-chip cache।

**2. Instruction Parallelism**

**Pipelining:** Assembly line-এর মতো → CPU একসাথে বিভিন্ন stage-এ instruction execute করে। প্রতিটি stage একই সময়ে আলাদা instruction নিয়ে কাজ করে, ফলে CPU দ্রুত কাজ করতে পারে।

**Superscalar:** একাধিক pipeline → independent instructions parallel execute হয়। একাধিক pipeline ব্যবহার করে একাধিক instruction একসাথে সম্পন্ন করা যায়, তাই execution আরও দ্রুত হয়।

### ****Limitations****

* Parallel execution এবং cache বৃদ্ধি এখন diminishing returns এ পৌঁছেছে।
* Clock rate বাড়ানো → power problem (heat)।

### ****Modern Approach: Multicore Processors****

* এক chip-এ multiple processors (cores) রাখা → performance বাড়ে clock rate বাড়ানো ছাড়া।
* Shared large cache → efficiency বাড়ায়।
* Software যদি effectively use করতে পারে → doubling cores ≈ doubling performance।
* Advantage: two simpler cores > one complex core (less power, better cache utilization)।

✅ **Summary:**

* Clock rate বাড়ানো এখন সীমিত।
* Cache এবং parallelism ব্যবহার করা হয়েছে।
* ভবিষ্যতে multiple cores (multicore) approachই প্রধান strategy।

ঠিক আছে, আমি Embedded Systems এবং ARM-এর বিষয়টা সহজ এবং সংক্ষেপে বোঝাচ্ছি:

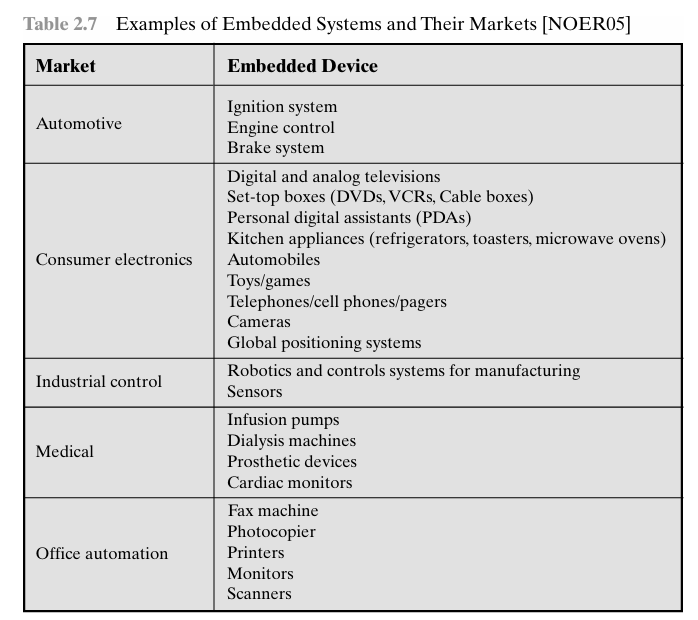
**2.4 Embedded Systems and the ARM (Self Reading)**

**Embedded Systems:**  
Embedded system হলো একটি বিশেষ উদ্দেশ্যে তৈরি কম্পিউটার সিস্টেম। এটি সাধারণ-purpose কম্পিউটার (যেমন ল্যাপটপ বা ডেস্কটপ) নয়, বরং কোনো বড় সিস্টেম বা প্রোডাক্টের অংশ হিসেবে কাজ করে। উদাহরণ: গাড়ির ABS (anti-lock braking system), ওয়াশিং মেশিন, মোবাইল ফোন।

**Embedded Systems:**  
An embedded system is a computer system designed for a specific purpose. Unlike general-purpose computers (such as laptops or desktops), it functions as a part of a larger system or product.  
**Example:** Car ABS (anti-lock braking system), washing machine, mobile phone.

**বিশেষ বৈশিষ্ট্য:**

* নির্দিষ্ট কাজের জন্য ডিজাইন করা।
* সাইজ, ব্যয়, গতি, নির্ভরযোগ্যতা ইত্যাদিতে ভিন্ন ভিন্ন সীমাবদ্ধতা থাকতে পারে।
* পরিবেশের সঙ্গে ঘনিষ্ঠভাবে যুক্ত থাকতে পারে (যেমন সেন্সর বা actuator)।
* রিয়েল-টাইম constraints থাকতে পারে: সফটওয়্যারকে ঠিক সময়মতো কাজ করতে হয়।



**Embedded system-এর উপাদান (Figure 2.13 অনুসারে):**

* **Processor & Memory** – মূল কম্পিউটিং।
* **Sensors & Actuators** – পরিবেশের সঙ্গে মিথস্ক্রিয়া।
* **Interfaces** – মানুষের সাথে যোগাযোগ বা অন্য ডিভাইসের সঙ্গে সংযোগ।
* **FPGA / ASIC / Software** – বিশেষ উদ্দেশ্যে কার্যকরী।
* **Auxiliary systems** – পাওয়ার, কুলিং, সেফটি।
* **Diagnostic port** – সমস্যার সমাধান ও মনিটরিং।

ঠিক আছে 👍 এখন আমি **ARM Evolution** অংশটা সহজভাবে ব্যাখ্যা করছি:

**ARM Evolution**

**ARM কী?**

* **ARM (Advanced RISC Machine)** হলো একটি **RISC-based (Reduced Instruction Set Computer)** প্রসেসর আর্কিটেকচার।
* **ARM Inc., Cambridge (UK)** এ ডিজাইন করা হয়।
* ARM কোম্পানি নিজেরা প্রসেসর তৈরি করে না; বরং আর্কিটেকচার ডিজাইন করে **manufacturers-দের লাইসেন্স দেয়।**
* বৈশিষ্ট্য:
  + ছোট সাইজ (small die size)
  + কম শক্তি খরচ (low power consumption)
  + উচ্চ গতি (high-speed)
  + Embedded systems এবং handheld devices-এ ব্যাপক ব্যবহার।

📱 **উদাহরণ:** iPod, iPhone, গেমস ডিভাইস, মোবাইল ফোন, PDAs, consumer electronics ইত্যাদি।  
আজকের দিনে ARM হলো বিশ্বের **সবচেয়ে বেশি ব্যবহৃত প্রসেসর আর্কিটেকচার।**

**ইতিহাস (Origin of ARM):**

1. **1980s – Acorn Computers (UK):**
   * BBC-এর জন্য *BBC Computer Literacy Project* এ নতুন microcomputer architecture বানানোর দায়িত্ব পায়।
   * এর মাধ্যমে **Acorn RISC Machine (ARM)** প্রসেসরের জন্ম।
2. **ARM1 (1985):**
   * প্রথম সংস্করণ।
   * মূলত গবেষণা ও BBC মেশিনে coprocessor হিসেবে ব্যবহৃত।
3. **ARM2 (1985):**
   * ARM1-এর তুলনায় বেশি ফিচার এবং গতি।
4. **ARM3 (1989):**
   * আরও উন্নত সংস্করণ।
5. **VLSI Technology:**
   * Acorn chip তৈরি করত না।
   * **VLSI Technology** ARM-এর fabrication করত এবং ARM বাজারজাত করার লাইসেন্স পায়।
   * ধীরে ধীরে অন্য কোম্পানিগুলো ARM-কে তাদের **embedded systems**-এ ব্যবহার শুরু করে।
6. **ARM Ltd (1990):**
   * Acorn, VLSI, এবং Apple মিলে নতুন কোম্পানি বানায়: **ARM Ltd.**
   * Acorn RISC Machine → **Advanced RISC Machine।**
   * প্রথম প্রোডাক্ট: **ARM6** (ARM3-এর উন্নত সংস্করণ)।
7. এরপর একের পর এক নতুন ARM family আসে → প্রতিবারই বেশি পারফরম্যান্স, বেশি ফিচার, কম পাওয়ার খরচ।

**ARM-এর ব্যবহার ক্ষেত্র (ARM official categories):**

ARM processors তিন ধরনের সিস্টেমের জন্য তৈরি হয়:

1. **Embedded real-time systems**
   * Storage systems
   * Automotive (গাড়ির body ও power-train control)
   * Industrial systems
   * Networking
2. **Application platforms**
   * Open OS (Linux, Palm OS, Symbian, Windows CE)
   * মোবাইল, wireless devices
   * Consumer electronics (TV, camera ইত্যাদি)
   * Digital imaging applications
3. **Secure applications**
   * Smart cards
   * SIM cards
   * Payment terminals

👉 **সংক্ষেপে:**  
ARM প্রসেসর শুরু হয়েছিল Acorn-এর RISC প্রকল্প হিসেবে। এখন এটি embedded systems থেকে মোবাইল পর্যন্ত সর্বত্র ব্যবহার হচ্ছে। ARM হলো ছোট, শক্তি-সাশ্রয়ী, দ্রুত এবং সস্তা প্রসেসরের আর্কিটেকচার, যার বাজারে আধিপত্য সবচেয়ে বেশি।

হ্যাঁ 👍 একদম ঠিক ধরেছো।

👉 **ARM মূলত প্রসেসরের আর্কিটেকচার ডিজাইন করে**, নিজেরা চিপ বানায় না।  
তারা কেবল **Instruction Set Architecture (ISA)** আর **মাল্টিকোর প্রসেসর আর্কিটেকচার** ডিজাইন করে এবং এগুলোর **লাইসেন্স বিক্রি করে**।

* যেমন, **Qualcomm (Snapdragon), Apple (A-series, M-series), Samsung (Exynos), MediaTek, Nvidia**— এরা সবাই ARM-এর কাছ থেকে আর্কিটেকচার লাইসেন্স নিয়ে নিজেদের প্রসেসর বানায়।
* ARM ডিজাইন দেয় → কোম্পানিগুলো সেই ডিজাইন ব্যবহার করে নিজেরা **ফ্যাব্রিকেশন (chip বানানো)** করে।

📌 তাই, **ARM = ডিজাইন কোম্পানি (architecture + ISA)**  
**Other companies = ARM ডিজাইন ব্যবহার করে প্রসেসর তৈরি করে।**

চাও তাহলে আমি ARM architecture লাইসেন্সের ধরন (যেমন Cortex license, architecture license) ব্যাখ্যা করে দিই?