(ISA) Instruction set architecture is the structure of a computer that a machine language program (or a compiler) must understand to write a correct program for that machine.

\[ \text{APPLICATION} \quad \text{OPERATING SYSTEM} \quad \text{COMPILER} \quad \text{PORTION OF ISA} \]

\[ \text{INSTRUCTION SET ARCHITECTURE} \quad \text{I/O SYSTEM} \]

\[ \text{DIGITAL DESIGN} \quad \text{CIRCUIT DESIGN} \]

\[ \text{CISC ARCHITECTURE WITH MICROPROGRAMMED CONTROL UNIT} \]

CISC stands for complex instruction set computing.

\[ \text{MAIN MEMORY} \quad \text{CACHE MEMORY} \quad \text{MICROPROGRAMMED CONTROL UNIT} \]

\[ \text{INSTRUCTION AND DATA PATH} \quad \text{CONTROL UNIT} \]

\[ \text{RISC ARCHITECTURE WITH HARDWARE CONTROL UNIT} \]

RISC stands for reduced instruction set computing.

\[ \text{MAIN MEMORY} \quad \text{INSTRUCTION CACHE MEMORY} \quad \text{DATA CACHE MEMORY} \]

\[ \text{HARWARE CONTROL UNIT} \quad \text{DATA PATH} \]

\[ \text{STAGES IN INSTRUCTION EXECUTION} \]

\[ \text{FETCH INSTRUCTION} \quad \text{DECODE INSTRUCTION} \quad \text{EXECUTE INSTRUCTION} \quad \text{WRITE-BACK} \]

(FI) (DI) (EI)
## Comparison between RISC and CISC

<table>
<thead>
<tr>
<th>S. No.</th>
<th>RISC</th>
<th>CISC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Simple instructions, few in numbers</td>
<td>Many complex instructions</td>
</tr>
<tr>
<td>2</td>
<td>Fixed length instructions</td>
<td>Variable length instructions</td>
</tr>
<tr>
<td>3</td>
<td>Few addressing modes (3-5)</td>
<td>Many addressing modes (12-24)</td>
</tr>
<tr>
<td>4</td>
<td>Only LOAD/STORE instructions access memory</td>
<td>Many instructions can access memory</td>
</tr>
<tr>
<td>5</td>
<td>Complexity in compiler</td>
<td>Complexity in microcode</td>
</tr>
<tr>
<td>6</td>
<td>Average CPI is less than 1.5</td>
<td>CPI is in between 2 and 15</td>
</tr>
<tr>
<td>7</td>
<td>Support large number of general purpose registers</td>
<td>Support 8 to 24 number of general purpose registers</td>
</tr>
<tr>
<td>8</td>
<td>Highly pipelined</td>
<td>Less pipelined</td>
</tr>
</tbody>
</table>

### CISC processor
- Can be built either with a single chip or with multiple chips mounted on a printed circuit board.

### RISC processor
- It moves from frequent operations into software, thus dedicating hardware resources to the most frequently used operations.

### Subclass of RISC processors
- RISC processors, which allow multiple instructions to be issued simultaneously during each cycle. Thus, the effective CPI of a superscalar processor should be lower than that of a scalar RISC processor. The clock rate of a superscalar processor matches that of scalar RISC processors.
Organizing memory and storage systems is known as memory hierarchy.

Memory hierarchy stores the information by satisfying three properties:

1. Inclusion - It is stated by the following set inclusion relations among memory levels:
   \[ M_1 \subseteq M_2 \subseteq M_3 \subseteq \ldots \subseteq M_n \]
   
   \( M_0 \rightarrow \) cache memory, (lowest level)
   \( M_n \rightarrow \) contains all of the information words stored (highest level).

2. Coherence - Copies of the same information item at higher levels of the memory hierarchy are consistent. Two strategies to maintain:
   - Write-Through (WF) - Update data immediately to all higher levels.
   - Write-Back (WB) - Delay the update of copies at higher levels until the data being modified in the lower level is replaced or removed from that level.

3. Locality - Three dimensions of locality principle are:
   - Temporal locality - If a program accesses one memory address, there is a good chance that it will access the same address again.
   - Spatial locality - If a program accesses one memory address, there is a good chance that it will also access other nearby addresses.
   - Sequential locality - Execution of program that follows a certain sequential order.
Memory capacity planning

→ Hit Rate →

When a needed item is found in the level of the memory hierarchy being examined, it is called a hit. Otherwise, it is called a miss.

\[ \text{Hit Rate} = h_i (\text{between 0 and 1}) \]
\[ \text{Miss Rate} = 1 - h_i \]

We assume, \( h_0 = 0 \) and \( h_n = 1 \).

Access frequency, \( f_i \) to level \( M_i = (1-h_1) \times (1-h_2) \times \ldots \times h_i \)

Note that \( \sum_{i=0}^{n} f_i = 1 \) and \( \sum_{i=0}^{n} f_i = 1 \).

→ Effective access time →

\[ T_{eff} = \sum_{i=1}^{n} t_i f_i = h_1 t_1 + (1-h_1) h_2 t_2 + \ldots + (1-h_1)(1-h_2) \ldots (1-h_{n-1}) h_n + n \]

→ Hierarchy optimization →

Cost of a memory hierarchy is estimated as follows:

\[ C_{total} = \sum_{i=1}^{n} C_i S_i \]

Since \( C_1 \geq C_2 \geq C_3 \geq \ldots \geq C_n \), we have to choose \( S_1 < S_2 < S_3 < \ldots < S_n \).

5. Interleaved memory organization –

In an interleaved memory, the memory is divided into a set of banks. An interleaved memory with \( n \) banks is said to be \( n \)-way interleaved.

There are two types of interleaving:

1. High-order
2. Low-order

(\( q \)-bits one high)
High-order interleaving:

Memory conflicts are easily avoided each process executes a different program and programs stored in separate memory modules. Interconnection network is set to connect each process to its proper memory module.

Low-order interleaving:

Consecutive memory locations reside in different memory modules. Process executing a program stored in a contiguous block of memory would need to access different modules simultaneously. Simultaneous access possible but difficult to avoid memory conflicts.

Disadvantages of interleaved memory:

1. Involves complex design
2. Reduced fault-tolerance
3. Cannot be expanded incrementally

Pipelined Memory Areas:

Memory Address Region (6 bits)

```
<table>
<thead>
<tr>
<th>Module Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>M0</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>16</td>
</tr>
<tr>
<td>24</td>
</tr>
<tr>
<td>32</td>
</tr>
<tr>
<td>40</td>
</tr>
<tr>
<td>48</td>
</tr>
<tr>
<td>56</td>
</tr>
</tbody>
</table>
```

8-Way interleaved memory
Timing of the pipelined access of the eight contiguous memory words:

\[ T = \frac{\Theta}{m} \]

\( \Theta = \) Major Cycle

\( I = \) Minor Cycle

\( m = \) Degree of interleaving

---

**Bandwidth and Fault Tolerance**

**Memory Bandwidth** (B)

B of m-way interleaved memory is upper bounded by m

Hellerman estimate of B, \[ B = m^{0.56} \]

Average vector access time \[ t_i = \frac{\Theta}{m} \left( 1 + \frac{m-1}{n} \right) \]

when \( n \rightarrow n \) element

\( m \rightarrow m \)-way lower order memory system

For long vectors, \[ t_i = \frac{\Theta}{m} = \Theta \]

For vector access, \( n = 1 \) and \( t_i = \Theta \)

**Fault Tolerance**

One non-memory module faults lead to the failure of the system is known as fault tolerance.

In a low order interleaved system, will parallelize

In a high order interleaved system, will be used.
VLIW architecture - (Very long Instruction Word)

- In a VLIW machine, many operations (instructions in a normal machine) are encoded in a single instruction. The word of the instruction is very long and can contain multiple operations independent.

```
     INSTRUCTION CACHE

     INSTRUCTION REGISTER

     EXECUTION UNIT #1  EXECUTION UNIT #2  EXECUTION UNIT #3  EXECUTION UNIT #4

     DATA CACHE
```

VLIW instruction execution -

```
add r1, r2, r3  load r4, r5+4  mov r6/2  mul r7, r8, r9
```

Advantages - simpler hardware, good compilers, compiler-friendly,
Run-time behaviour is highly predictable, Easily managed exceptions & interrupts

Disadvantages - larger no. of registers, larger code size

Backplane Bus System -

Backplane Bus Specification -

```
<table>
<thead>
<tr>
<th>CPU BOARD</th>
<th>MEMORY BOARD</th>
<th>BUS CONTROLLER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slot 1</td>
<td>Slot k-1</td>
<td>Slot k</td>
</tr>
</tbody>
</table>
```

BACKPLANE (signal lines and connectors)

DATA TRANSFER BUS (DTB)

- DTB ARBITRATION BUS
- Interrupt and Synchronization Bus
- UTILITY BUS
**Addressing and Timing Protocols**

**Bus Addressing**

The backplane bus is driven by a digital clock with a fixed cycle time called a cycle. Backplane has limited physical size, so it will not store information.

**NOT** bus transactions have one slave/master.

**Broadcast** - Read operation where multiple slaves share data on the bus.

**Inter-transaction** - Read operation involving multiple slaves. It implements multi-cache coherence on the bus.

---

**Bus Timing Protocols**

<table>
<thead>
<tr>
<th>Synchronous Bus</th>
<th>Asynchronous Bus</th>
</tr>
</thead>
<tbody>
<tr>
<td>It includes a clock in the control lines and a fixed protocol for sending address and data relative to the clock.</td>
<td>Handshaking protocols are used between the sender and receiver.</td>
</tr>
<tr>
<td>These buses can be both fast and inexpensive.</td>
<td>Buses are slow.</td>
</tr>
<tr>
<td>Buses cannot be long.</td>
<td>Buses can be long enough.</td>
</tr>
<tr>
<td>Everything on a bus must run at the same clock rate.</td>
<td>Some clocks are not used.</td>
</tr>
<tr>
<td>CPU-memory links are typically synchronous.</td>
<td>I/O links are more likely to be asynchronous.</td>
</tr>
</tbody>
</table>

---

**Arbitration - Process of selecting next bus master**

**Central Arbitration**

```
<table>
<thead>
<tr>
<th></th>
<th>Req. 1</th>
<th>Potential master 1</th>
<th>Potential master 2</th>
<th>Potential master n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event</td>
<td>Central</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Req. 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Event</td>
<td>Central</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Req. 3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Event</td>
<td>Central</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

**Daisy Chain**
Distributed (Decentralized) arbitration - primarily local arbitration

→ Transaction mode -

1. Address only transfer (Address)
2. Compelled - data transfer (Address + 1 to 2 blocks of data)
3. Paced - data transfer (Address + fixed length block of data)

A bus transaction consists of a request followed by a response.

→ Interrupt Mechanism -

It is a request from I/O to a process for service or attention.