|  |  |  |
| --- | --- | --- |
|  | **Hardwired Control Unit** | **Micro-programmed Control Unit** |
| Implementation | Fixed set of logic gates and circuits | Microcode stored in memory |
| Flexibility | Less flexible, difficult to modify | More flexible, easier to modify |
| Instruction Set | Supports limited instruction sets | Supports complex instruction sets |
| Complexity of Design | Simple design, easy to implement | Complex design, more difficult to implement |
| Speed | Fast operation | Slower operation due to microcode decoding |
| Debugging and Testing | Difficult to debug and test | Easier to debug and test |
| Size and Cost | Smaller size, lower cost | Larger size, higher cost |
| Maintenance and Upgradability | Difficult to upgrade and maintain | Easier to upgrade and maintain |

|  |  |
| --- | --- |
| **SRAM** | **DRAM** |
| It stores information as long as the power is supplied. | It stores information as long as the power is supplied or a few milliseconds when the power is switched off. |
| Transistors are used to store information in SRAM. | Capacitors are used to store data in DRAM. |
| Capacitors are not used hence no refreshing is required. | To store information for a longer time, the contents of the capacitor need to be refreshed periodically. |
| SRAM is faster compared to DRAM. | DRAM provides slow access speeds. |
| It does not have a refreshing unit. | It has a refreshing unit. |
| These are expensive. | These are cheaper. |
| SRAMs are low-density devices. | DRAMs are high-density devices. |
| In this bit are stored in voltage form. | In this bit is stored in the form of electric energy. |
| These are used in cache memory. | These are used in main memory. |
| Consumes less power and generates less heat. | Uses more power and generates more heat. |
| SRAMs has lower latency | DRAM has more latency than SRAM |
| SRAMs are more resistant to radiation than DRAM | DRAMs are less resistant to radiation than SRAMs |
| SRAM has higher data transfer rate | DRAM has lower data transfer rate |
| SRAM is used in high-speed cache memory | DRAM is used in lower-speed main memory |
| SRAM is used in high performance applications | DRAM is used in general purpose applications |

### ****ADD (Addition)****

**Description:** Adds the source operand to the destination operand and stores the result in the destination.

**Example:**

assembly

MOV AL, 05H ; AL = 05H

MOV BL, 03H ; BL = 03H

ADD AL, BL ; AL = AL + BL = 05H + 03H = 08H

**Flags Affected:**

* **CF (Carry Flag)** – Set if unsigned overflow occurs.
* **PF (Parity Flag)** – Set if the result has even parity.
* **AF (Auxiliary Flag)** – Set if there’s a carry from bit 3 to bit 4.
* **ZF (Zero Flag)** – Set if the result is zero.
* **SF (Sign Flag)** – Set if the result is negative (MSB = 1).
* **OF (Overflow Flag)** – Set if signed overflow occurs.

### ****ADC (Add with Carry)****

**Description:** Adds the source operand, destination operand, and the **Carry Flag (CF)**, storing the result in the destination.

**Example:**

assembly

MOV AX, 0FFFFH ; AX = FFFFH

MOV BX, 0001H ; BX = 0001H

STC ; Set Carry Flag (CF = 1)

ADC AX, BX ; AX = AX + BX + CF = FFFFH + 0001H + 1 = 0001H (with CF=1)

**Flags Affected:** Same as **ADD**.

### ****INC (Increment)****

**Description:** Increments the operand by **1**. Does **not** affect the **Carry Flag (CF)**.

**Example:**

assembly

MOV CX, 00FFH

INC CX ; CX = CX + 1 = 0100H

**Flags Affected:**

* **PF, AF, ZF, SF, OF** (CF remains unchanged).

### ****SUB (Subtraction)****

**Description:** Subtracts the source operand from the destination operand and stores the result in the destination.

**Example:**

assembly

MOV AL, 0AH ; AL = 0AH (10)

MOV BL, 03H ; BL = 03H (3)

SUB AL, BL ; AL = AL - BL = 0AH - 03H = 07H

**Flags Affected:** Same as **ADD**, but CF indicates borrow.

### ****SBB (Subtract with Borrow)****

**Description:** Subtracts the source operand, destination operand, and the **Carry Flag (CF)**, storing the result in the destination.

**Example:**

assembly

MOV AX, 1234H

MOV BX, 5678H

STC ; Set Carry Flag (CF = 1)

SBB AX, BX ; AX = AX - BX - CF = 1234H - 5678H - 1 = (result depends)

**Flags Affected:** Same as **SUB**.

### ****DEC (Decrement)****

**Description:** Decrements the operand by **1**. Does **not** affect the **Carry Flag (CF)**.

**Example:**

assembly

MOV DL, 01H

DEC DL ; DL = DL - 1 = 00H

**Flags Affected:**

* **PF, AF, ZF, SF, OF** (CF remains unchanged).

### ****NEG (Negate)****

**Description:** Computes the **two’s complement** of the operand (inverts sign).

**Example:**

assembly

MOV AL, 05H

NEG AL ; AL = -5 (FBH in 2's complement)

**Flags Affected:**

* **CF** – Set unless the operand was zero.
* **PF, AF, ZF, SF, OF** (modified based on result).

### ****CMP (Compare)****

**Description:** Compares two operands by subtracting them **without storing the result**, only updating flags.

**Example:**

assembly

MOV AL, 0AH

CMP AL, 05H ; Sets flags as (AL - 05H)

**Flags Affected:** Same as **SUB**.

### ****DAS (Decimal Adjust after Subtraction)****

**Description:** Adjusts the result of a **BCD subtraction** to ensure correct packed BCD format.

**Example:**

assembly

MOV AL, 35H ; AL = 35H (BCD 35)

SUB AL, 09H ; AL = 2CH (invalid BCD)

DAS ; Adjusts AL to 26H (BCD 26)

**Flags Affected:**

* **CF, AF, PF, ZF, SF** (modified based on adjustment).

### ****MUL (Unsigned Multiplication)****

**Description:** Multiplies **AL** (8-bit) or **AX** (16-bit) by the operand, storing the result in **AX** or **DX:AX**.

**Example (8-bit):**

assembly

MOV AL, 05H

MOV BL, 03H

MUL BL ; AX = AL \* BL = 000FH

**Flags Affected:**

* **CF & OF** – Set if result exceeds register size.
* **SF, ZF, AF, PF** – Undefined.

### ****IMUL (Signed Multiplication)****

**Description:** Performs **signed multiplication** (similar to **MUL** but preserves sign).

**Example:**

assembly

MOV AL, -2 ; AL = FEH (2's complement)

MOV BL, 3 ; BL = 03H

IMUL BL ; AX = AL \* BL = FFFAH (-6)

**Flags Affected:** Same as **MUL**.

### ****DIV (Unsigned Division)****

**Description:** Divides **AX** (16-bit) or **DX:AX** (32-bit) by the operand, storing quotient in **AL/AX** and remainder in **AH/DX**.

**Example (8-bit):**

assembly

MOV AX, 000FH ; AX = 000FH (15)

MOV BL, 03H ; BL = 03H (3)

DIV BL ; AL = 05H (quotient), AH = 00H (remainder)

**Flags Affected:**

* **All flags undefined after DIV**.

### ****IDIV (Signed Division)****

**Description:** Performs **signed division** (similar to **DIV** but preserves sign).

**Example:**

assembly

MOV AX, -15 ; AX = FFF1H (-15 in 2's complement)

MOV BL, 3 ; BL = 03H

IDIV BL ; AL = FBH (-5), AH = 00H

**Flags Affected:**

* **All flags undefined after IDIV**.

### ****Direct Mapping****

* **Definition**: Each block of main memory maps to **exactly one cache line** (no flexibility).
* **Mapping Formula**:
* Cache Line = (Main Memory Block Address) MOD (Number of Cache Lines)
* **Advantages**:
  + Simple and fast (easy to implement in hardware).
  + Low search time (only one possible location for a block).
* **Disadvantages**:
  + **High conflict misses** (if multiple blocks map to the same cache line, frequent replacements occur).
  + Poor cache utilization (even if cache has free space, a block may be forced to replace an existing one).

### ****Fully Associative Mapping****

* **Definition**: Any main memory block can be placed in **any cache line** (maximum flexibility).
* **Mapping**: No fixed mapping; cache controller searches all lines for a match.
* **Advantages**:
  + **Low conflict misses** (blocks don’t compete for the same cache line).
  + Better cache utilization (can store any block anywhere).
* **Disadvantages**:
  + **High search time** (requires checking all cache lines for a match).
  + Complex hardware (needs a **Content-Addressable Memory (CAM)** for parallel lookups).
  + Expensive to implement for large caches.

### ****Set-Associative Mapping****

* **Definition**: A compromise between direct and fully associative mapping.
* **Mapping**: Cache is divided into **sets**, and each set contains multiple lines (ways).

 Set Number = (Main Memory Block Address) MOD (Number of Sets)

* A block can be placed in **any line within its mapped set**.

 **Types**:

* **2-way set-associative** (2 lines per set).
* **4-way set-associative**, etc.

 **Advantages**:

* **Reduces conflict misses** compared to direct mapping.
* Faster than fully associative (only search within a set).
* More practical for large caches.

 **Disadvantages**:

* More complex than direct mapping.
* Slower than direct mapping (due to set search).

**Capacity**

* Refers to the amount of data a memory unit can hold (measured in bytes, kilobytes, megabytes, gigabytes, etc.).
* Examples: RAM (4GB–128GB), Hard Drives (500GB–10TB+).

**2. Access Time**

* The time taken to read from or write to memory.
* **Faster access** in cache & RAM vs. **slower access** in HDDs.
* Measured in **nanoseconds (ns)** for RAM, **milliseconds (ms)** for HDDs.

**3. Volatility**

* **Volatile Memory**: Loses data when power is off (e.g., RAM, Cache).
* **Non-Volatile Memory**: Retains data without power (e.g., ROM, SSDs, HDDs).

**4. Cost (Price per Bit)**

* Faster memory (e.g., Cache, RAM) is more expensive than slower storage (e.g., HDDs).
* Memory hierarchy balances speed and cost.

**5. Physical Type**

* **Semiconductor Memory** (e.g., RAM, ROM, Flash).
* **Magnetic Memory** (e.g., HDDs).
* **Optical Memory** (e.g., CDs, DVDs).

**6. Read/Write Ability**

* **Read-Only Memory (ROM)**: Permanent data (e.g., firmware).
* **Read-Write Memory (RAM)**: Temporary data storage for active processes.

**7. Location in Hierarchy**

* **Primary Memory (Main Memory)**: Fast, directly accessed by CPU (e.g., RAM).
* **Secondary Memory**: Slower, used for long-term storage (e.g., SSDs, HDDs).
* **Cache Memory**: Extremely fast, stores frequently used data near CPU.

**8. Transfer Rate (Bandwidth)**

* Speed at which data moves between memory and CPU (measured in MB/s or GB/s).
* Example: DDR5 RAM has higher bandwidth than DDR4.

**1. Immediate Addressing Mode**

* **Operand is part of the instruction itself.**  
  **Example:** MOV AX, 1234H (Loads 1234H directly into AX).

**2. Register Addressing Mode**

* **Operands are in registers.**  
  **Example:** MOV BX, AX (Copies value from AX to BX).

**3. Direct Addressing Mode**

* **Memory address is given directly in the instruction.**  
  **Example:** MOV AX, [5000H] (Loads data from memory location 5000H into AX).

**4. Register Indirect Addressing Mode**

* **Memory address is stored in a register (BX, SI, DI, BP).**  
  **Example:** MOV AX, [BX] (Loads data from memory location pointed by BX into AX).