⚙️ STM32 Inline Assembly (Thumb Mode) Cheat Sheet

🧠 Works in **STM32CubeIDE** using `__asm volatile()`
Processor: **Cortex-M3 (STM32F103C8)**
Mode: **Thumb-2**

🟢 1️⃣ Basic Syntax

```
__asm volatile ("instruction");
```

একাধিক লাইন লিখতে:

```
__asm volatile (
    "MOV R0, #5\n"
    "MOV R1, #10\n"
    "ADD R2, R0, R1\n"
);
```

🔹 2️⃣ Data Movement (Move / Load / Store)

Assembly	অর্থ	C Equivalent
`MOV R0, #10`	Immediate মান R0 তে দাও	`int x = 10;`
`MOV R1, R0`	এক রেজিস্টার থেকে আরেকটায় কপি	`b = a;`
`LDR R0, =0x4001100C`	রেজিস্টারে constant address লোড	pointer assign
`LDR R1, [R0]`	মেমোরি থেকে মান লোড	`R1 = *R0;`
`STR R1, [R0]`	মেমোরিতে মান স্টোর	`*R0 = R1;`

🔹 3️⃣ Arithmetic Operations

Assembly	কাজ	C Equivalent
`ADD R0, R1, R2`	R0 = R1 + R2	`a = b + c;`
`SUB R0, R0, #1`	R0 = R0 - 1	`a--;`
`MUL R0, R1, R2`	R0 = R1 × R2	`a = b * c;`
`UDIV R0, R1, R2`	R0 = R1 ÷ R2	`a = b / c;`
`AND R0, R1, R2`	Bitwise AND	`a = b & c;`
`ORR R0, R1, R2`	Bitwise OR	`a = b
`EOR R0, R1, R2`	Bitwise XOR	`a = b ^ c;`

🔹 8⃣ Branching / Loop

Assembly	কাজ
`B label`	Unconditional jump
`BEQ label`	যদি Zero flag = 1 হয়
`BNE label`	যদি Zero flag = 0 হয়
`CMP R0, R1`	Compare R0 এবং R1
`BL function`	Function call (Branch with Link)
`BX LR`	Function থেকে ফিরে আসা

🔹 ⏲️ Delay Loop Example

```
void delay(void)
{
    __asm volatile (
        "ldr r0, =1000000\n"
        "1:\n"
        "subs r0, r0, #1\n"
        "bne 1b\n"
    );
}
```

🔹 🔌 GPIO Operation (STM32 Blue Pill)

🟩 A. GPIO Initialization (PC13 as output)

```
__asm volatile (
    "ldr r0, =0x40021018\n"   // RCC_APB2ENR
    "ldr r1, [r0]\n"
    "orr r1, r1, #(1 << 4)\n" // Enable GPIOC clock
    "str r1, [r0]\n"

    "ldr r0, =0x40011004\n"   // GPIOC_CRH
    "ldr r1, [r0]\n"
    "bic r1, r1, #(0xF << 20)\n"
    "orr r1, r1, #(0x1 << 20)\n" // Output mode
    "str r1, [r0]\n"
);
```

🟦 B. GPIO Set / Reset (LED Toggle)

```
__asm volatile (
    "ldr r0, =0x4001100C\n"  // GPIOC_ODR
    "ldr r1, [r0]\n"
    "eor r1, r1, #(1 << 13)\n" // Toggle PC13
    "str r1, [r0]\n"
);
```

🟨 **C. Combine with Delay**

```
void asm_led_toggle(void)
{
    __asm volatile (
        "ldr r0, =0x4001100C\n"   // GPIOC_ODR
        "ldr r1, [r0]\n"
        "eor r1, r1, #(1 << 13)\n"
        "str r1, [r0]\n"

        "ldr r2, =1000000\n"      // Delay
        "1:\n"
        "subs r2, r2, #1\n"
        "bne 1b\n"
    );
}
```

🔹 9️⃣ Function Integration

```
int main(void)
{
    // Enable GPIOC Clock
    RCC->APB2ENR |= RCC_APB2ENR_IOPCEN;
    GPIOC->CRH &= ~(0xF << 20);
    GPIOC->CRH |=  (0x1 << 20);

    while (1)
    {
        asm_led_toggle();
    }
}
```

🔹 🔢 Bit Operations Cheat Table

Operation	Assembly	C Equivalent
Set bit	`orr r0, r0, #(1<<n)`	`reg

Clear bit	`bic r0, r0, #(1<<n)`	`reg &= ~(1<<n);`
Toggle bit	`eor r0, r0, #(1<<n)`	`reg ^= (1<<n);`
Test bit	`tst r0, #(1<<n)`	`if(reg & (1<<n))`

🔹 📡 Tips

- ✅ Always use `__asm volatile` → prevent compiler optimization
- ✅ Use `ldr` for any constant > 255
- ⚠️ Don't forget `BX LR` in naked assembly functions
- ⚙️ Build option: **-O0** (no optimization) for proper delay timing
- 🧠 Use `1b` and `1f` for backward/forward label refere