



# **ARM Microcontroller Internship Programme**

**Trainer: Kiran Jaybhave**

**Sunbeam – Industrial Training Programme**

**Sunbeam Pune (Hinjawadi)**



- **Prerequisites**
  - **C Programming:**
    - Proficiency in C syntax, pointers, memory management, structures, bit manipulation, and preprocessor directives
  - **Basic Electronics:**
    - Understanding of digital electronics, analog circuits, voltage levels, timing diagrams, and basic circuit analysis

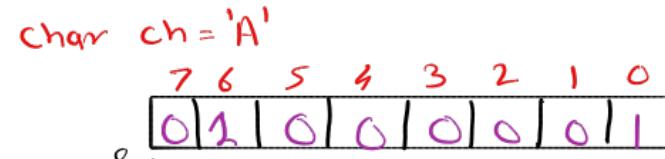


# Bit Manipulation

(65)

## • What is Bit Manipulation?

- Operations performed directly on bits (0s and 1s).
- Used heavily in **embedded systems, register programming, and I/O control.**



## • Why Bit Manipulation is Important?

- Used for **GPIO control, interrupt flags, status registers.**
- Essential for **memory-mapped I/O and hardware-level programming.**
- Helps reduce code size and improves speed.

Q	R
2	65
2	32
2	16
2	8
2	4
2	2
2	1
0	2



## • Common Bit Operations

1. **Set a bit** → Make a bit = 1  
value |= (1 << n);
2. **Clear a bit** → Make a bit = 0  
value &= ~(1 << n);
3. **Toggle a bit** → Flip bit (0→1 or 1→0)  
value ^= (1 << n);
4. **Check a bit** → Test if bit = 1  
if (value & (1 << n))
5. **Masking** → Isolate selected bits  
value & mask
6. **Shifting** → Move bits left or right  
value << n, value >> n



# Print Num in Binary,

-10  $\Rightarrow$  True

10  $\Rightarrow$  True

0  $\Rightarrow$  False

num = 10

num  $\Rightarrow$ 

0	0	0	0	1	1	0	.	0
---	---	---	---	---	---	---	---	---

$\& 100$

Void \*vp = 100;  
size = 2;  $\Rightarrow$

char mask = 0x 80;

1	0	0	0	0	0	0
---	---	---	---	---	---	---

void print-bin (Void \*vp, int size)  
{  
 mask = 0x 80;  
 while (mask)  
 {  
 if (mask & \* (char\*) vp)  
 pf (1);  
 else  
 pf (0);  
 mask = mask >> 1;  
 }  
}

mask = 

0	0	0	0	1	0	0	0
---	---	---	---	---	---	---	---

  
VP  $\&$ 

0	0	0	0	1	0	1	0
---	---	---	---	---	---	---	---

0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

 $\Rightarrow 0$   

0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

 $\Rightarrow 0$   

0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

 $\Rightarrow 0$   

0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

 $\Rightarrow 0$   

0	0	0	0	1	0	0	0
---	---	---	---	---	---	---	---

 $\Rightarrow 8 \Rightarrow 1$



int regv = 0xFAFA;

00  
0000 0000

00  
0000 0000

FA

FA

1111 1010

1111 1010

mask = 0X80

103

mask = 0X80

102

mask = 0X80

101

mask = 0X80

100



## Set a bit

# define

BV(n)

$1 \ll n$

- Turn a specific bit to 1 without changing other bits.

$\text{Reg}_r = 0x0A;$



$\begin{array}{cccccc} 7 & 6 & 5 & 4 & 3 & 2 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ \end{array}$

→ set 5<sup>th</sup> bit

$\begin{array}{cccccc} 7 & 6 & 5 & 4 & 3 & 2 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ \end{array}$

formula:  $\text{Reg}_r = \text{Reg}_r | \text{BV}(n)$

OR

$\text{Reg}_r |= \text{BV}(n)$

$\begin{array}{cccccc} 7 & 6 & 5 & 4 & 3 & 2 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ \end{array} \Rightarrow \text{Reg}_r$

$\begin{array}{cccccc} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ \end{array} \Rightarrow \underline{\text{BV}(5)}$

$\begin{array}{cccccc} 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \end{array} \rightarrow \begin{array}{cccccc} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ \end{array} \quad (1 \ll 5)$

# ➤ Clear a bit

# define BVCS)  $1 \ll 5$

- Makes a specific bit 0 without affecting other bits.

$\text{Regr} = 0xFA;$

$\begin{array}{r} 7654 \quad 3210 \\ 1111 \quad 1010 \end{array}$

$\rightarrow$  Clear 5<sup>m</sup> bit

$\begin{array}{r} 7654 \quad 3210 \\ 1101 \quad 1010 \end{array}$

Formula:

$$\text{Regr} = \text{Regr} \& \sim(\text{BV}(n))$$

$$\text{Regr} \& S = \sim(\text{BV}(n))$$

$\begin{array}{r} 7654 \quad 3210 \\ \rightarrow 1111 \quad 1010 \\ 0010 \quad 0000 \end{array}$

$\text{BV}(5)$

$\sim \text{BV}(5)$

$\begin{array}{r} 7654 \quad 3210 \\ \rightarrow 1101 \quad 1010 \\ 0010 \quad 0000 \end{array}$

$\text{BV}(5)$

$\begin{array}{r} 7654 \quad 3210 \\ \rightarrow 1101 \quad 1010 \\ \sim \text{BV}(5) \end{array}$

$\begin{array}{r} 7654 \quad 3210 \\ 1101 \quad 1010 \\ \hline 1101 \quad 1010 \end{array}$

$\begin{array}{r} 7654 \quad 3210 \\ 0000 \quad 0000 \\ \hline 0010 \quad 0000 \end{array}$

## ➤ Toggle a bit

- Flips the bit value (0 becomes 1, and 1 becomes 0).

$$\underline{\text{Regr}} = 0 * FA$$

7	6	5	4	3	2	1	0
1	1	1	1	1	0	1	0

toggle the 3<sup>rd</sup> bit ( $1 \rightarrow 0 / 0 \rightarrow 1$ )

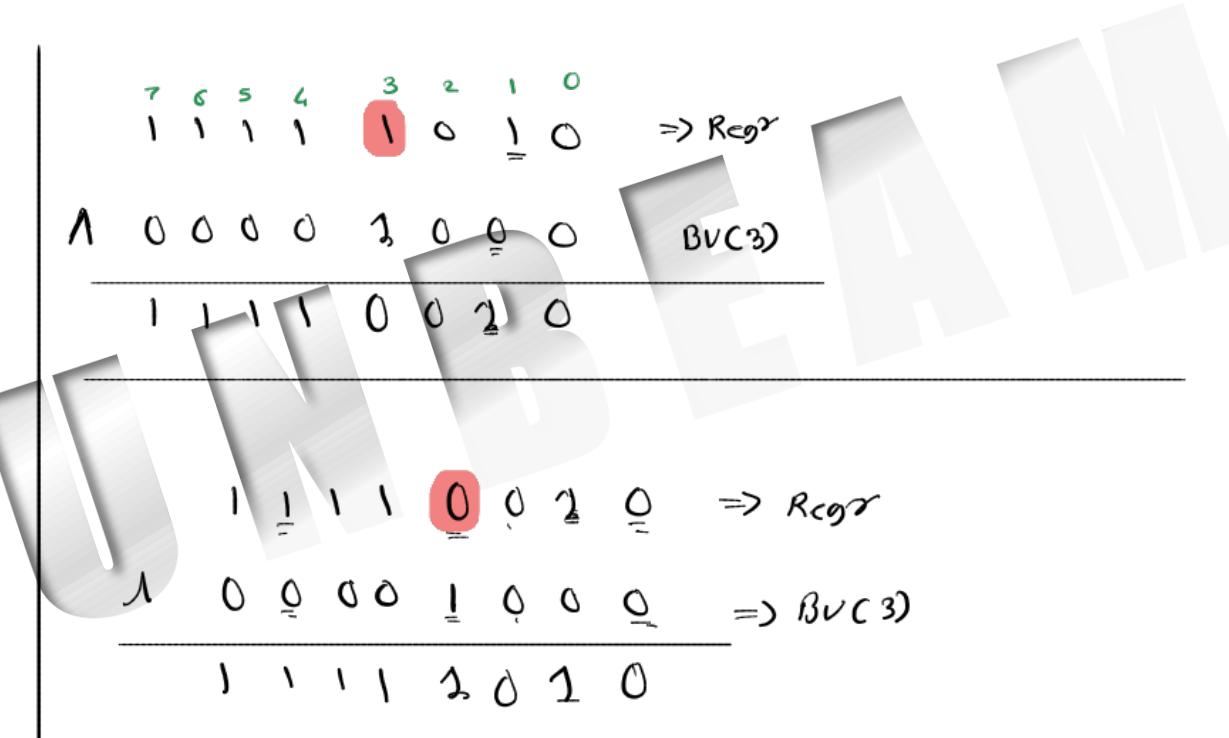
7	6	5	4	3	2	1	0
1	1	1	1	0	0	1	0

$$\text{Regr} = \underline{\text{Regr}} \wedge \text{Bv}(n)$$

OR

$$\text{Regr} \wedge \text{Bv}(n)$$

# S



## ➤ Check a bit

- Tests whether a specific bit is 1 or 0.

$\text{Regr} = 0x FA$

[ 7 6 5 4 3 2 1 0 ]  
[ 1 1 1 1 1 0 1 0 ]

Check 2<sup>nd</sup> bit is set or clear

if ( $\text{Regr}$  &  $BV(2)$ )

2<sup>nd</sup> bit is set

else

→ 2<sup>nd</sup> bit is clear

8 7 6 5 4 3 2 1 0  
0 0 0 0 0 1 0 1 0 0  
0 0 0 0 0 0 0 0 0 0  
 $\Rightarrow 0 \Rightarrow \text{False}$

## ➤ Masking

---

- Extracts only the required bits and hides the rest.

SUNBEAM



## ➤ Left Shifting

- Moves bits to the left, multiplying the value by powers of two.

SUNBEAM



## ➤ Right Shifting

- Moves bits to the right, dividing the value by powers of two.

SUNBEAM



# ➤ Read value from bit $n^{th}$ to $n^{th}$ of register

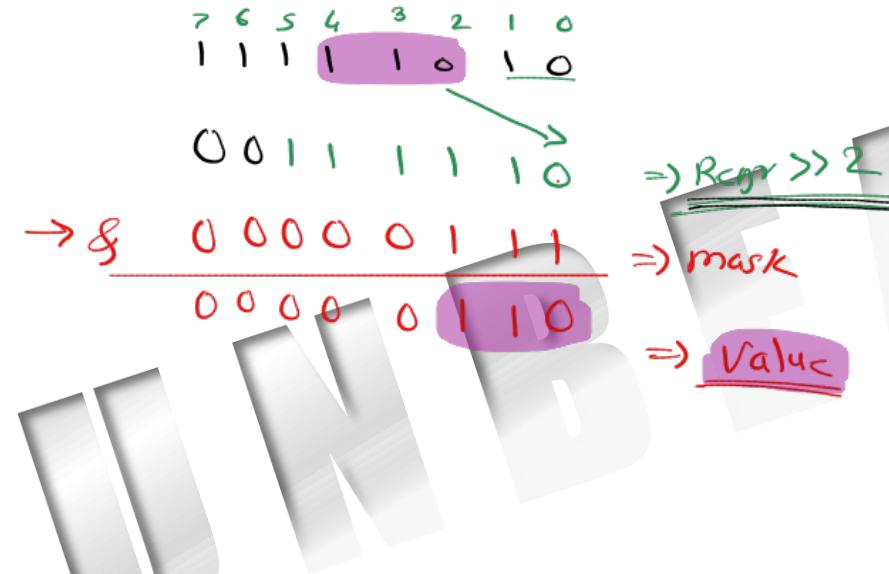
$$Regr = 0 \times FA$$

7 6 5 4 3 2 1 0  
 1 1 1 1 1 0 1 0  
 → extract value of bit  $\underline{\min}$  to  $\underline{\max}$   
 $(110) = \underline{6}$

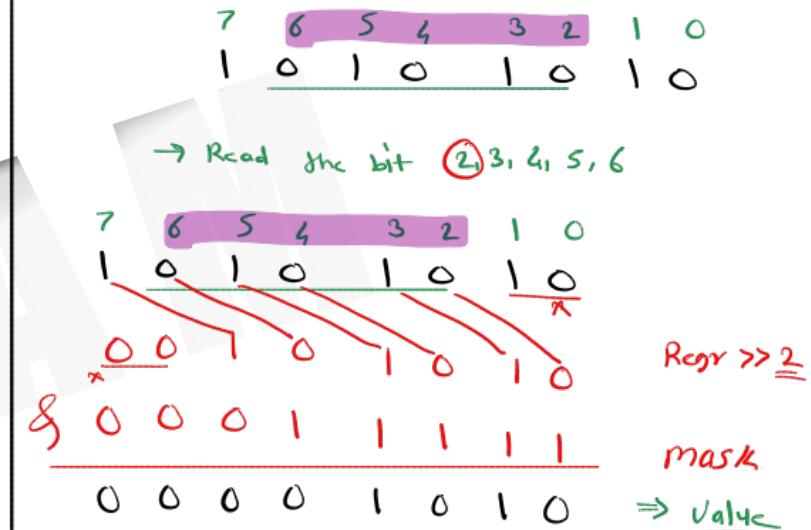
$$Value = (Regr \gg \underline{\min}) \underline{\&} \underline{mask}$$

$$\begin{aligned}
 mask &= (\max - \min) + 1 \\
 &= (4 - 2) + 1 \\
 &\Rightarrow 2 + 1 \\
 &\Rightarrow \underline{3} \quad (111)
 \end{aligned}$$

SUNB



$$Regr = 0x A A$$



## ➤ Write value from bit $n^{th}$ to $n^{th}$ of register

$$Regr = 0 * FA$$

  
1 1 1 1 1 0 1 0

→ extract value of bit  $\frac{min}{2,3,4}$ ,  $\frac{max}{4}$   
 $(000) = 0$

Formula :-

$$Regr = \underline{Regr} \& \sim(Bv(2) | Bv(\underline{3}) | Bv\underline{(4)})$$



  
1 1 1 1 1 0 1 0

$$0 0 0 0 \quad 0 1 0 0 \Rightarrow Bv(2)$$

$$0 0 0 0 \quad 1 0 0 0 \Rightarrow Bv(3)$$

$$0 0 0 1 \quad 0 0 0 0 \Rightarrow Bv(4)$$

$$\underline{0 0 0 1} \quad 1 1 0 0 0 \quad (Bv(2) | Bv(\underline{3}) | Bv\underline{(4)})$$

$$\underline{0 0 0 1} \quad 1 1 0 0 0 \quad \sim(Bv(2) | Bv(\underline{3}) | Bv\underline{(4)})$$

$$1 1 1 \underline{0 0 0} \quad 1 0 \quad \leftarrow$$

## \* Polling to bit (Check bit continuously)

$R_{ngr} = [ \begin{smallmatrix} 7 & 6 & 5 & 4 \\ 0 & 0 & 0 & 0 \end{smallmatrix} \quad \begin{smallmatrix} 3 \\ 0 \end{smallmatrix} \quad \begin{smallmatrix} 2 & 1 & 0 \\ 0 & 0 & 0 \end{smallmatrix} ]$

while ( $(R_{ngr} \& BV(3)) == 0$ )

;



$$\begin{array}{r} 0 & 0 & 0 & 0 & \begin{smallmatrix} 0 \\ 0 \end{smallmatrix} & 0 & 0 & 0 \\ \& 0 & 0 & 0 & | & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \Rightarrow \underline{\underline{0}}$$

$R_{ngr} = [ \begin{smallmatrix} 7 & 6 & 5 & 4 \\ 0 & 0 & 0 & 0 \end{smallmatrix} \quad \begin{smallmatrix} 3 \\ 1 \end{smallmatrix} \quad \begin{smallmatrix} 2 & 1 & 0 \\ 0 & 0 & 0 \end{smallmatrix} ]$

while ( $(R_{ngr} \& BV(3)) == 1$ )

;



$$\begin{array}{r} 0 & 0 & 0 & 0 & \begin{smallmatrix} 1 \\ 1 \end{smallmatrix} & 0 & 0 & 0 \\ \& 0 & 0 & 0 & | & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{array} \Rightarrow \begin{array}{l} \text{Non-} \\ \text{zero} \\ (\text{loop}) \end{array}$$

$\Rightarrow 0$   
 (Stop the loop)





**Thank you!**  
Kiran Jaybhave  
email – [kiran.jaybhave@sunbeaminfo.com](mailto:kiran.jaybhave@sunbeaminfo.com)

