Opcode → Operation Table

1. Arithmetic Operations

Opcode	Operation name	What it does	Uses
0	ADD	`RESULT = A + B`	`ALU_ADD`
1	SUB	`RESULT = A - B`	`ALU_SUB`
2	INC	`RESULT = A + 1`	`ALU_INC`
3	DEC	`RESULT = A - 1`	`ALU_DEC`
4	ADC (Add with carry)	`RESULT = A + B + C_in` (C_in from `FLAGS` bit 0)	`ALU_ADC`
5	SBB (Sub with borrow)	`RESULT = A - B - Borrow` (from `FLAGS` bit 0)	`ALU_SBB`

2. Logical Operations

Opcode	Operation name	What it does	Uses
6	AND	`RESULT = A AND B`	`ALU_AND`
7	OR	`RESULT = A OR B`	`ALU_OR`
8	XOR	`RESULT = A XOR B`	`ALU_XOR`
9	NOT	`RESULT = NOT A`	`ALU_NOT`
10	NAND	`RESULT = NOT (A AND B)`	`ALU_NAND`
11	NOR	`RESULT = NOT (A OR B)`	`ALU_NOR`

3. Shift / Rotate Operations

Opcode	Operation name	What it does	Uses
12	LSL (Logical left)	`RESULT = A << (B & 0xF)`	`SHIFT_LSL`
13	LSR (Logical right)	`RESULT = A >> (B & 0xF)` (zero-fill)	`SHIFT_LSR`
14	ASL (Arithmetic left)	`RESULT = A << (B & 0xF)`	`SHIFT_ASL`
15	ASR (Arithmetic right)	`RESULT = A >>_arith (B & 0xF)` (sign-ext)	`SHIFT_ASR`
16	ROL (Rotate left)	`RESULT = (A<<n) | (A>>(16−n))`, `n=B&0xF`	`SHIFT_ROL`
17	ROR (Rotate right)	`RESULT = A ROR (B & 0xF)`	`SHIFT_ROR`

4. Barrel Shifter

Opcode	**Operation name**	**What it does**	**Uses**
18	BARREL_SHIFT	`RESULT = A << (B & 0xF)`	`BARREL_SHIFT`

5. Sequential Multiplication (Booth placeholder)

Opcode	**Operation name**	**What it does**	**Uses**
19	MUL / Booth	`RESULT = A * B`	`BOOTH_MUL`

6. Comparator Unit

Opcode	Operation name	Output location	Encoding in `CMP_OUT`	Uses
20	COMPARE	`CMP_OUT`	`0001` (1) → A > B	`COMPARATOR` + `CMP_GT`
			`0010` (2) → A == B	`CMP_EQ`
			`0100` (4) → A < B	`CMP_LT`

7. Parity Checker

Opcode	Operation name	What it checks	Where stored	Uses
21	PARITY	Parity of all bits of **A**	`RESULT`	`PARITY_CHECK`

- Loop over bits of `A`, flipping a parity bit.
- At the end, `RESULT = 1` means **even parity**, `RESULT = 0` means **odd parity**.