QCOM CPU Architecture Specification

This document provides the official technical specification for the QCOM CPU architecture. It serves as the definitive reference for software developers, compiler engineers, and system designers working with the QCOM instruction set and memory model. The QCOM architecture is an 8-bit processor featuring a paged memory model, specifically designed for embedded systems, real-time control applications, and simple computing tasks where efficiency and deterministic performance are paramount.

1.0 Core Architecture

This section details the fundamental components of the QCOM architecture, including its memory organization, register set, and status flags. A thorough understanding of these core elements is crucial for effective low-level programming, system optimization, and the development of supporting tools like compilers and debuggers. We will begin with an examination of the paged memory model.

1.1 Memory Model

The QCOM CPU utilizes a paged memory architecture to access a larger address space than would typically be possible with 8-bit offsets. An instruction provides an 8-bit offset, which is combined with a 4-bit page number to form a 12-bit effective address. This model allows the CPU to address up to 4096 bytes (4 KB) of memory.

The active 4-bit page number is sourced from the upper four bits (bits 7-4) of the special-purpose `R7` register. The final 12-bit effective address is calculated as follows:

```
effective_address = ((R7 >> 4) & 0x0F) << 8 | (offset & 0xFF)
```

The memory space has several reserved regions for specific functions, such as video display and I/O.

Memory Map

Address Range	Size	Purpose
`0x000 - 0x07F`	128 bytes	Display Buffer (Video RAM)
`0x080`	1 byte	Input Controller State (Memory-Mapped I/O)

`0x090`	1 byte	Program Start Address (Default Program Counter)

1.2 Registers

The QCOM CPU is equipped with eight 8-bit general-purpose registers, denoted `R0` through `R7`. While `R0` through `R6` are fully general-purpose, the `R7` register serves a dual purpose, acting as both a standard register and a system control register. This design conserves the limited register file space, a common practice in 8-bit architectures, by combining status and memory control into a single register.

The two functions of `R7` are detailed below:

- **Status Flags**: The least significant bit of `R7` serves as the system's primary status flag.
    - **Bit 0: Zero Flag (ZF)**: The Zero Flag is set to `1` if the result of an arithmetic, logical, shift, or move operation is exactly zero. It is cleared to `0` otherwise. This flag is the basis for all conditional control flow instructions.
- **Memory Page Selector**: The most significant four bits of `R7` hold the active page number for memory access.
    - **Bits 7-4: Page Number**: These four bits provide the page value (`0x0` to `0xF`) used in the calculation of all 12-bit effective memory addresses.

2.0 Operand Types and Addressing Modes

Instructions operate on data using a strictly defined set of operand types and addressing modes. This section defines the syntax for these operands as used in QCOM assembly language and details the methods by which the CPU locates and accesses the data required for an operation. The subsequent subsections will define the specific operand types.

2.1 Operand Types

The QCOM assembly language supports four distinct operand types, each with a specific syntax and purpose. These types determine whether an instruction uses a register, a constant value, or a memory address as a data source or destination.

Type	Syntax	Description
REG	`R[0-7]`	An 8-bit general-purpose register, specified by its number (e.g., `R0`, `R3`).

IMM	`$[value]`	An 8-bit immediate value embedded directly within the instruction's bytecode.
ADDR	`[value]`	An 8-bit address offset. The value can be represented in decimal, hexadecimal (`0x...`), or binary (`0b...`).
LABEL	`#[name]`	A symbolic name representing a memory address. This is resolved by the compiler into a final address value.

2.2 Addressing Modes

Addressing modes define how the CPU interprets an instruction's operands to locate the data it needs to operate on. QCOM supports five fundamental addressing modes.

3.0 Instruction Set Reference

This section provides a comprehensive reference for the entire QCOM instruction set. For clarity, instructions are grouped by their primary function: data transfer, I/O, bitwise manipulation, logic, arithmetic, and control flow. Each entry specifies the instruction's mnemonic, its corresponding opcode(s) and syntax, a description of its operation, and its effect on the Zero Flag (ZF). We will begin with data transfer instructions.

3.1 Data Transfer Instructions

These instructions are used to move data between registers, memory, and immediate values. While `MOV` provides fundamental register-memory transfers, the `MIL` and `MFI` instructions enable more advanced data structures and pointer-based algorithms by introducing register-indirect and memory-indirect addressing.

Mnemonic	Opcode	Syntax	Bytes	Description	Flags Affected
`MOV`	`0x10`	`MOV REG, IMM`	2	Moves an 8-bit immediate value into a register.	ZF is set based on the result.
`MOV`	`0x11`	`MOV ADDR, REG`	2	Stores the 8-bit value from a register into a memory location.	ZF is set based on the result.

MOV	0x12	MOV REG, ADDR	2	Loads an 8-bit value from a memory location into a register.	ZF is set based on the result.
MOV	0x13	MOV REG, REG	2	Copies an 8-bit value from a source register to a destination register.	ZF is set based on the result.
MIL	0x50	MIL REG, IMM	2	Stores an immediate value at the memory address pointed to by a register.	ZF is set based on the value of the source register (REG), not the immediate value being moved.
MIL	0x51	MIL REG, REG	2	Stores a value from a source register at the memory address pointed to by another register.	ZF is set based on the value of the destination address register, not the value being moved.
MIL	0x52	MIL REG, ADDR	2	Stores a value from a memory location at the address pointed to by a register.	ZF is set based on the value of the destination address register, not the value being moved.
MIL	0x53	MIL ADDR, REG	2	Stores a register's value at the address pointed to by a value in memory.	ZF is set based on the final value written to the destination memory location.
MFI	0x54	MFI REG, REG	2	Loads a value from the memory address pointed to by a source register into a destination register.	ZF is set based on the result.
MFI	0x55	MFI REG, ADDR	2	Loads a value from an indirect memory location (pointed to by another memory location) into a register.	ZF is set based on the result.
MFI	0x56	MFI ADDR, REG	2	Stores a value from the memory address pointed to by a register into another memory location.	ZF is set based on the result.

`MFI`	`0x57`	`MFI ADDR, ADDR`	2	Stores a value from an indirect memory location (pointed to by a memory location) into another memory location.	ZF is set based on the result.

3.2 I/O and System Instructions

These instructions manage interactions with external devices, control the display, and halt the CPU.

Mnemonic	Opcode	Syntax	Bytes	Description	Flags Affected
`DIS`	`0x01`	`DIS IMM`	2	Sets an internal display register to an immediate value for debugging.	None
`DIS`	`0x02`	`DIS REG`	2	Sets an internal display register to the value of a register.	None
`DIS`	`0x03`	`DIS ADDR`	2	Sets an internal display register to the value from a memory location.	None
`IN`	`0x04`	`IN REG`	2	Reads the 8-bit state from the input controller at `0x080` into a register.	ZF
`OUT`	`0x05`	`OUT IMM, IMM`	3	Sends an immediate value to an I/O port specified by an immediate value.	None
`OUT`	`0x06`	`OUT IMM, REG`	3	Sends a register's value to an I/O port specified by an immediate value.	None
`OUT`	`0x07`	`OUT IMM, ADDR`	3	Sends a value from memory to an I/O port specified by an immediate value.	None
`BRK`	`0x0F`	`BRK`	1	Halts program execution.	None
`SHW`	`0x14`	`SHW`	1	Renders the contents of the display buffer (`0x000-0x07F`) to the screen.	None

CLS	0x15	CLS IMM	2	Clears the screen to a specific color value.	None

3.3 Bitwise Shift and Rotate Instructions

These instructions perform bitwise shifts and rotations on operands in registers or memory. A logical shift moves bits and fills the vacated position with zero, while a rotate moves bits and wraps the shifted-out bit to the other end of the byte.

Mnemonic	Opcode	Syntax	Bytes	Description	Flags Affected
SBL	0x18	SBL REG	2	Performs a logical shift left by one bit on a register.	ZF
SBL	0x19	SBL ADDR	2	Performs a logical shift left by one bit on a value in memory.	ZF
SBR	0x1A	SBR REG	2	Performs a logical shift right by one bit on a register.	ZF
SBR	0x1B	SBR ADDR	2	Performs a logical shift right by one bit on a value in memory.	ZF
RBL	0x1C	RBL REG	2	Rotates a register's bits left by one position.	ZF
RBL	0x1D	RBL ADDR	2	Rotates the bits of a value in memory left by one position.	ZF
RBR	0x1E	RBR REG	2	Rotates a register's bits right by one position.	ZF
RBR	0x1F	RBR ADDR	2	Rotates the bits of a value in memory right by one position.	ZF

3.4 Logical Instructions

These instructions perform bitwise logical operations (AND, OR, XOR, NOT). All instructions in this group affect the Zero Flag based on the result of the operation.

Mnemonic	Opcode	Syntax	Bytes	Description	Flags Affected
AND	0x20	AND REG, IMM	2	Performs a bitwise AND between a register and an immediate value.	ZF
AND	0x21	AND ADDR, REG	2	Performs a bitwise AND between a memory location and a register value.	ZF
AND	0x22	AND REG, ADDR	2	Performs a bitwise AND between a register and a memory value.	ZF
AND	0x23	AND REG, REG	2	Performs a bitwise AND between two registers.	ZF
OR	0x24	OR REG, IMM	2	Performs a bitwise OR between a register and an immediate value.	ZF
OR	0x25	OR ADDR, REG	2	Performs a bitwise OR between a memory location and a register value.	ZF
OR	0x26	OR REG, ADDR	2	Performs a bitwise OR between a register and a memory value.	ZF
OR	0x27	OR REG, REG	2	Performs a bitwise OR between two registers.	ZF
XOR	0x28	XOR REG, IMM	2	Performs a bitwise XOR between a register and an immediate value.	ZF
XOR	0x29	XOR ADDR, REG	2	Performs a bitwise XOR between a memory location and a register value.	ZF

XOR	0x2A	XOR REG, ADDR	2	Performs a bitwise XOR between a register and a memory value.	ZF
XOR	0x2B	XOR REG, REG	2	Performs a bitwise XOR between two registers.	ZF
NOT	0x2C	NOT REG	2	Performs a bitwise NOT (inversion) on a register.	ZF
NOT	0x2D	NOT ADDR	2	Performs a bitwise NOT (inversion) on a value in memory.	ZF

3.5 Arithmetic Instructions

These instructions perform 8-bit integer arithmetic. Operations wrap on overflow (i.e., they are performed modulo 256). All instructions in this group affect the Zero Flag based on the 8-bit result.

Mnemonic	Opcode	Syntax	Bytes	Description	Flags Affected
ADD	0x30	ADD REG, IMM	2	Adds an immediate value to a register.	ZF
ADD	0x31	ADD ADDR, REG	2	Adds a register value to a memory location.	ZF
ADD	0x32	ADD REG, ADDR	2	Adds a memory value to a register.	ZF
ADD	0x33	ADD REG, REG	2	Adds the value of one register to another.	ZF
SUB	0x34	SUB REG, IMM	2	Subtracts an immediate value from a register.	ZF

SUB	0x35	SUB ADDR, REG	2	Subtracts a register value from a memory location.	ZF
SUB	0x36	SUB REG, ADDR	2	Subtracts a memory value from a register.	ZF
SUB	0x37	SUB REG, REG	2	Subtracts the value of one register from another.	ZF
INC	0x38	INC REG	2	Increments a register's value by one.	ZF
INC	0x39	INC ADDR	2	Increments a memory location's value by one.	ZF
DEC	0x3A	DEC REG	2	Decrements a register's value by one.	ZF
DEC	0x3B	DEC ADDR	2	Decrements a memory location's value by one.	ZF

3.6 Control Flow Instructions

These instructions alter the flow of program execution by changing the program counter (PC), either unconditionally or based on the state of the Zero Flag.

Mnemonic	Opcode	Syntax	Bytes	Description	Flags Affected
JMP	0x40	JMP IMM	2	Unconditionally jumps to the specified address offset.	None
JMP	0x41	JMP REG	2	Unconditionally jumps to the address contained in a register.	None
JIF	0x42	JIF IMM, IMM	3	Jumps to the target address if the condition operand is 1 AND the Zero Flag is set (ZF=1). If the condition operand is 0, the jump is never taken.	None

`JIF`	`0x43`	`JIF IMM, REG`	3	Jumps to the address in a register if the condition operand is `1` AND the Zero Flag is set (`ZF=1`). If the condition operand is `0`, the jump is never taken.	None
`JNI`	`0x44`	`JNI IMM, IMM`	3	Jumps to the target address if the condition operand is `1` AND the Zero Flag is clear (`ZF=0`). If the condition operand is `0`, the jump is never taken.	None
`JNI`	`0x45`	`JNI IMM, REG`	3	Jumps to the address in a register if the condition operand is `1` AND the Zero Flag is clear (`ZF=0`). If the condition operand is `0`, the jump is never taken.	None

3.7 Stack Instructions

Stack operations are not implemented in the current version of the QCOM architecture.

4.0 Appendix: Complete Opcode Map

This appendix provides a consolidated map of all implemented QCOM opcodes, sorted numerically for quick reference during debugging and tool development.

Opcode (Hex)	Mnemonic	Operands
`0x01`	`DIS`	`IMM`
`0x02`	`DIS`	`REG`
`0x03`	`DIS`	`ADDR`
`0x04`	`IN`	`REG`
`0x05`	`OUT`	`IMM, IMM`
`0x06`	`OUT`	`IMM, REG`
`0x07`	`OUT`	`IMM, ADDR`

0x0F	BRK	None
0x10	MOV	REG, IMM
0x11	MOV	ADDR, REG
0x12	MOV	REG, ADDR
0x13	MOV	REG, REG
0x14	SHW	None
0x15	CLS	IMM
0x18	SBL	REG
0x19	SBL	ADDR
0x1A	SBR	REG
0x1B	SBR	ADDR
0x1C	RBL	REG
0x1D	RBL	ADDR
0x1E	RBR	REG
0x1F	RBR	ADDR
0x20	AND	REG, IMM
0x21	AND	ADDR, REG
0x22	AND	REG, ADDR
0x23	AND	REG, REG
0x24	OR	REG, IMM
0x25	OR	ADDR, REG
0x26	OR	REG, ADDR
0x27	OR	REG, REG

0x28	XOR	REG, IMM
0x29	XOR	ADDR, REG
0x2A	XOR	REG, ADDR
0x2B	XOR	REG, REG
0x2C	NOT	REG
0x2D	NOT	ADDR
0x30	ADD	REG, IMM
0x31	ADD	ADDR, REG
0x32	ADD	REG, ADDR
0x33	ADD	REG, REG
0x34	SUB	REG, IMM
0x35	SUB	ADDR, REG
0x36	SUB	REG, ADDR
0x37	SUB	REG, REG
0x38	INC	REG
0x39	INC	ADDR
0x3A	DEC	REG
0x3B	DEC	ADDR
0x40	JMP	IMM
0x41	JMP	REG
0x42	JIF	IMM, IMM
0x43	JIF	IMM, REG
0x44	JNI	IMM, IMM

0x45	JNI	IMM, REG
0x50	MIL	REG, IMM
0x51	MIL	REG, REG
0x52	MIL	REG, ADDR
0x53	MIL	ADDR, REG
0x54	MFI	REG, REG
0x55	MFI	REG, ADDR
0x56	MFI	ADDR, REG
0x57	MFI	ADDR, ADDR