ARM Assembly Opcodes and Formats (Simplified)

This document provides a simplified overview of the ARM Assembly instruction opcodes and their basic encoding formats supported by this assembler. Please note that this is a highly simplified representation for educational purposes, and a real ARM architecture has much more complex and varied instruction formats.

General ARM Instruction Format (Simplified Data Processing)

Most data processing instructions (like ADD, SUB, AND, ORR, EOR) follow a general format:

`[Cond] [Opcode] [I] [S] [Rn] [Rd] [Operand2]`

- `Cond` (bits 31-28): Condition code (e.g., `1110` for AL - Always, `1101` for LE - Less than or Equal, etc.). Our assembler currently defaults to `1110` (AL) for most instructions unless specified for branches.
- `Opcode` (bits 27-24): Defines the instruction type (e.g., `0010` for ADD, `0100` for SUB).
- `I` (bit 25): Immediate bit. `1` if `Operand2` is an immediate value, `0` if `Operand2` is a register.
- `S` (bit 20): S-bit. `1` if the instruction updates condition flags (e.g., CMP, TST), `0` otherwise. Our assembler implicitly sets this for CMP.
- `Rn` (bits 19-16): First operand register.
- `Rd` (bits 15-12): Destination register.
- `Operand2` (bits 11-0): Second operand. Can be an immediate value or a shifted register.

Instruction Opcodes and Encoding Examples

MOV (Move)

- **`MOV Rd, #imm`** : Move Immediate to Register
  - **Opcode Base**: `0xE3A00000` (for `MOV` with immediate)
  - **Format**: `1110 0011 1010 Rd Rd Rd Rd 0000 0000 Imm Imm Imm Imm`
  - `Rd` : Destination Register (bits 15-12)
  - `Imm` : 8-bit immediate value (bits 7-0)
  - **Example**: `MOV R0, #10` -> `E3A0000A`
- **`MOV Rd, Rm`** : Move Register to Register
  - **Opcode Base**: `0xE1A00000` (for `MOV` with register)
  - **Format**: `1110 0001 1010 Rd Rd Rd Rd 0000 0000 0000 Rm Rm Rm Rm`
  - `Rd` : Destination Register (bits 15-12)
  - `Rm` : Source Register (bits 3-0)
  - **Example**: `MOV R9, R0` -> `E1A09000`

ADD (Add)

- **`ADD Rd, Rn, #imm`** : Add Immediate to Register
  - **Opcode Base**: `0xE2800000`
  - **Format**: `1110 0010 1000 Rn Rn Rn Rn Rd Rd Rd Rd 0000 0000 Imm Imm Imm Imm`
  - `Rn` : First Source Register (bits 19-16)
  - `Rd` : Destination Register (bits 15-12)
  - `Imm` : 8-bit immediate value (bits 7-0)
  - **Example**: `ADD R2, R0, #3` -> `E2802003`

- **`ADD Rd, Rn, Rm`** : Add Register to Register
  - **Opcode Base**: `0xE0800000`
  - **Format**: `1110 0000 1000 Rn Rn Rn Rn Rd Rd Rd Rd 0000 0000 0000 Rm Rm Rm Rm`
  - `Rn` : First Source Register (bits 19-16)
  - `Rd` : Destination Register (bits 15-12)
  - `Rm` : Second Source Register (bits 3-0)
  - **Example**: `ADD R2, R0, R1` -> `E0802001`

SUB (Subtract)

- **`SUB Rd, Rn, #imm`** : Subtract Immediate from Register
  - **Opcode Base**: `0xE2400000`
  - **Format**: `1110 0010 0100 Rn Rn Rn Rn Rd Rd Rd Rd 0000 0000 Imm Imm Imm Imm`
  - `Rn` : First Source Register (bits 19-16)
  - `Rd` : Destination Register (bits 15-12)
  - `Imm` : 8-bit immediate value (bits 7-0)
  - **Example**: `SUB R3, R2, #5` -> `E2423005`
- **`SUB Rd, Rn, Rm`** : Subtract Register from Register
  - **Opcode Base**: `0xE0400000`
  - **Format**: `1110 0000 0100 Rn Rn Rn Rn Rd Rd Rd Rd 0000 0000 0000 Rm Rm Rm Rm`
  - `Rn` : First Source Register (bits 19-16)
  - `Rd` : Destination Register (bits 15-12)
  - `Rm` : Second Source Register (bits 3-0)

- **Example**: `SUB R3, R2, R0` -> `E0423000`

CMP (Compare)

- **`CMP Rn, #imm`** : Compare Register with Immediate
  - **Opcode Base**: `0xE3500000`
  - **Format**: `1110 0011 0101 Rn Rn Rn Rn 0000 0000 0000 0000 Imm Imm Imm Imm`
  - `Rn` : Register to compare (bits 19-16)
  - `Imm` : 8-bit immediate value (bits 7-0)
  - **Example**: `CMP R3, #10` -> `E353000A`
- **`CMP Rn, Rm`** : Compare Register with Register
  - **Opcode Base**: `0xE1500000`
  - **Format**: `1110 0001 0101 Rn Rn Rn Rn 0000 0000 0000 0000 Rm Rm Rm Rm`
  - `Rn` : First Register to compare (bits 19-16)
  - `Rm` : Second Register to compare (bits 3-0)
  - **Example**: `CMP R3, R0` -> `E1530000`

AND (Logical AND)

- **`AND Rd, Rn, #imm`** : Logical AND with Immediate
  - **Opcode Base**: `0xE2000000`
  - **Format**: `1110 0010 0000 Rn Rn Rn Rn Rd Rd Rd Rd 0000 0000 Imm Imm Imm Imm`
  - `Rn` : First Source Register (bits 19-16)
  - `Rd` : Destination Register (bits 15-12)
  - `Imm` : 8-bit immediate value (bits 7-0)

- **Example**: `AND R6, R0, #2` -> `E2006002`

- **`AND Rd, Rn, Rm`**: Logical AND with Register
  - **Opcode Base**: `0xE0000000`
  - **Format**: `1110 0000 0000 Rn Rn Rn Rn Rd Rd Rd Rd 0000 0000 0000 Rm Rm Rm Rm`
  - `Rn`: First Source Register (bits 19-16)
  - `Rd`: Destination Register (bits 15-12)
  - `Rm`: Second Source Register (bits 3-0)
  - **Example**: `AND R6, R0, R1` -> `E0006001`

ORR (Logical OR)

- **`ORR Rd, Rn, #imm`**: Logical OR with Immediate
  - **Opcode Base**: `0xE3800000`
  - **Format**: `1110 0011 1000 Rn Rn Rn Rn Rd Rd Rd Rd 0000 0000 Imm Imm Imm Imm`
  - `Rn`: First Source Register (bits 19-16)
  - `Rd`: Destination Register (bits 15-12)
  - `Imm`: 8-bit immediate value (bits 7-0)
  - **Example**: `ORR R7, R1, #1` -> `E3817001`
- **`ORR Rd, Rn, Rm`**: Logical OR with Register
  - **Opcode Base**: `0xE1800000`
  - **Format**: `1110 0001 1000 Rn Rn Rn Rn Rd Rd Rd Rd 0000 0000 0000 Rm Rm Rm Rm`
  - `Rn`: First Source Register (bits 19-16)
  - `Rd`: Destination Register (bits 15-12)

- `Rm` : Second Source Register (bits 3-0)
- **Example**: `ORR R7, R1, R2` -> `E1817002`

EOR (Exclusive OR)

- **`EOR Rd, Rn, #imm`** : Exclusive OR with Immediate
  - **Opcode Base**: `0xE2200000`
  - **Format**: `1110 0010 0010 Rn Rn Rn Rn Rd Rd Rd Rd 0000 0000 Imm Imm Imm Imm`
  - `Rn` : First Source Register (bits 19-16)
  - `Rd` : Destination Register (bits 15-12)
  - `Imm` : 8-bit immediate value (bits 7-0)
  - **Example**: `EOR R8, R2, #4` -> `E2228004`
- **`EOR Rd, Rn, Rm`** : Exclusive OR with Register
  - **Opcode Base**: `0xE0200000`
  - **Format**: `1110 0000 0010 Rn Rn Rn Rn Rd Rd Rd Rd 0000 0000 0000 Rm Rm Rm Rm`
  - `Rn` : First Source Register (bits 19-16)
  - `Rd` : Destination Register (bits 15-12)
  - `Rm` : Second Source Register (bits 3-0)
  - **Example**: `EOR R8, R2, R3` -> `E0228003`

LSL (Logical Shift Left)

- **`LSL Rd, Rm, #imm`** : Logical Shift Left by Immediate
  - **Opcode Base**: `0xE1A00000` (same as MOV, distinguished by shift bits)
  - **Format**: `1110 0001 1010 Rd Rd Rd Rd Imm Imm Imm Imm Imm 0000 Rm Rm Rm Rm`

- `Rd` : Destination Register (bits 15-12)
- `Rm` : Source Register (bits 3-0)
- `Imm` : 5-bit shift amount (bits 11-7)
- **Example**: `LSL R9, R0, #1` -> `E1A09080`

LSR (Logical Shift Right)

- **`LSR Rd, Rm, #imm`** : Logical Shift Right by Immediate
  - **Opcode Base**: `0xE1A00000` (same as MOV, distinguished by shift bits)
  - **Format**: `1110 0001 1010 Rd Rd Rd Rd Imm Imm Imm Imm Imm 0001 Rm Rm Rm Rm`
  - `Rd` : Destination Register (bits 15-12)
  - `Rm` : Source Register (bits 3-0)
  - `Imm` : 5-bit shift amount (bits 11-7)
  - **Example**: `LSR R10, R1, #1` -> `E1A0A0A1`

MUL (Multiply)

- **`MUL Rd, Rm, Rs`** : Multiply
  - **Opcode Base**: `0xE0000090`
  - **Format**: `1110 0000 0000 Rd Rd Rd Rd Rs Rs Rs Rs 1001 Rm Rm Rm Rm`
  - `Rd` : Destination Register (bits 19-16)
  - `Rm` : First Source Register (bits 3-0)
  - `Rs` : Second Source Register (bits 11-8)
  - **Example**: `MUL R11, R0, R1` -> `E00B0190`

Branch Instructions (B, BGE, BLT)

- **`B label`** : Unconditional Branch
  - **Opcode Base**: `0xEA000000`
  - **Format**: `1110 1010 Offset Offset Offset Offset`
  - `Offset` : 24-bit signed offset, shifted left by 2 (bits 23-0). Calculated relative to PC.
  - **Example**: `B end_program` -> `EAFFFFFF` (assuming `end_program` is 1 instruction back)
- **`BGE label`** : Branch if Greater Than or Equal
  - **Opcode Base**: `0xDA000000`
  - **Format**: `1101 1010 Offset Offset Offset Offset`
  - `Offset` : Same as `B` .
  - **Example**: `BGE label_high` -> `DA000001`
- **`BLT label`** : Branch if Less Than
  - **Opcode Base**: `0xBA000000`
  - **Format**: `1011 1010 Offset Offset Offset Offset`
  - `Offset` : Same as `B` .
  - **Example**: `BLT loop` -> `BAFFFFFC`

SWI (Software Interrupt)

- **`SWI #imm`** : Software Interrupt
  - **Opcode Base**: `0xEF000000`
  - **Format**: `1110 1111 Imm Imm Imm Imm Imm Imm Imm Imm Imm Imm Imm Imm`
  - `Imm` : 24-bit immediate value (bits 23-0).
  - **Example**: `SWI #0` -> `EF000000`

Disclaimer: The opcodes and formats provided here are highly simplified for the purpose of this basic assembler. Real ARM architecture has more complex encoding rules, condition codes, and addressing modes. This document serves as a guide for the specific implementation of this assembler.