**ARM Instructions**

The left column is a description of the instruction and the right column shows the mnemonic with necessary arguments. <dest> stands for destination, where the result of an instruction will go. <suffix> is an optional modifier that can be added to most instructions. <Operand1> and <Operand2> are arguments that are manipulated by the instruction. Note that <Operand1> must always be a register, but <Operand2> may be a constant or shifted value. A constant (immediate) in ARM is signified by the pound sign (#) followed by the constant. <src> is a source (register) only used in the store instruction.

**Arithmetic Instructions:**

Add ADD <suffix> <dest>, <Operand1>, <Operand2>

Subtract SUB <suffix> <dest>, <Operand1>, <Operand2>

Reverse Subtract RSB <suffix> <dest>, <Operand1>, <Operand2>

Multiply\* MUL <suffix> <dest>, <Operand1>, <Operand2>

Multiply with Add\* MLA <suffix> <dest>, <Operand1>, <Operand2>, <Operand3>

\* <Operand1> and <Operand2> must be in a register

**Logical Instructions:**

Bitwise AND AND <suffix> <dest>, <Operand1>, <Operand2>

Bitwise OR ORR <suffix> <dest>, <Operand1>, <Operand2>

Bitwise EOR(XOR) EOR <suffix> <dest>, <Operand1>, <Operand2>

Bit Clear BIC <suffix> <dest>, <Operand1>, <Operand2>

Bitwise shifts in ARM can act as single instructions or they can be added to <Operand2> of another instruction. When using the latter bitwise shift, the shift instruction will be followed by a constant in a range depending on the shift instruction.

Logical Shift Left LSL <suffix> <dest>, <Operand1>, <Operand2>

or <instruction><suffix>, …, <Operand2>, LSL #(0-31)

Logical Shift Right LSR <suffix> <dest>, <Operand1>, <Operand2>

or <instruction><suffix>, …, <Operand2>, LSR #(1-32)

Arithmetic Shift Right ASR <suffix> <dest>, <Operand1>, <Operand2>

or <instruction><suffix>, …, <Operand2>, ASR #(1-32)

Rotate Right ROR <suffix> <dest>, <Operand1>, <Operand2>

or <instruction><suffix>, …, <Operand2>, ROR #(1-31)

**Branch, Compare, and Conditionals**

<label> stands for a marked position in the program. This is where the program will branch.

Branch B <suffix> <label>

Branch with Link BL <suffix> <label>

Compare\* CMP <suffix> <Operand1>, <Operand2>

\*CMP sets the status register flags and enable the use of the conditionals below.

Conditionals in ARM are represented as suffixes that follow an instruction. For example:  
BEQ loop % Branch if equal to

ADDLT R0, R1, R2 % Add if less than

Equal EQ

Not Equal NE

Always AL

Never NV

Greater than or Equal GE

Less than or Equal LE

Greater than GT

Less than LT

\* There are more conditionals than listed. You can find these in the ARM documentation.

**Memory Instructions:**

Move MOV <suffix> <dest>, <Operand2>

Load Register LDR <suffix> <dest>, [<Operand1>]

Store Register STR <suffix> <src>, [<Operand1>]

Load Multiple\* LDM <suffix>, <Operand1>, {<Registers>}

Store Multiple\* STM <suffix>, <Operand1>, {<Registers>}

\* <Registers> is a list of registers to load data to or store data from. Registers will be separated with a comma, and consecutive registers can be indicated with a dash e.g., R1-R3.

**Assembler Directives\*:**

“the following goes in the data segment” .data

“the following goes in the text segment” .text

Create a string .ascii “<string>”

Create a string followed by 0 .asciz “<string>”

Create a byte .byte <byte value>

Create 32-bit word .word <word value>

\* There are more directives than listed