Quiz 1 Handout - Modified EDSACjr ISA

Here is the ISA for a slightly modified version of the EDSACjr.

In the notation used in the table below, M[x] stands for the contents of the memory location addressed by x. Accum refers to the accumulator. ← signifies that data is transferred (copied) from the location to the right of the ← to the location on the left. The immediate variable n is an address or a literal depending on the context. The EDSACjr architecture allows programmers to put constants at any point in the memory when a program is loaded. All addresses are word addresses.

Opcode	Description	Bit Representation
ADD n	Accum ← Accum + M[n]	00001 n
SUB n	Accum ← Accum - M[n]	00010 n
LD n	Accum ← M[n]	00011 n
ST n	M[n] ← Accum	00100 n
CLEAR	Accum ← 0	00101 00000000000
OR n	Accum ← Accum | M[n]	00110 n
AND n	Accum ← Accum & M[n]	00111 n
MUL n	Accum ← Accum * M[n]	01000 n
SHIFTR n	Accum ← Accum >> M[n]	01001 n
SHIFTL n	Accum ← Accum << M[n]	01010 n
BGE n	If Accum ≥ 0 then PC ← n	01011 n
BLT n	If Accum < 0 then PC ← n	01100 n
END	Halt machine	01101 00000000000

The shifts are arithmetic shifts. All words are 16 bits long. As in EDSAC, instructions are encoded as integers. The first 5 bits are the opcode and the last 11 bits form the immediate field (an 11-bit immediate address addresses up to 2048 words (16-bit) of memory -- twice that of the real EDSAC). Integers are represented in 16 bits, the most significant bit being a sign bit.