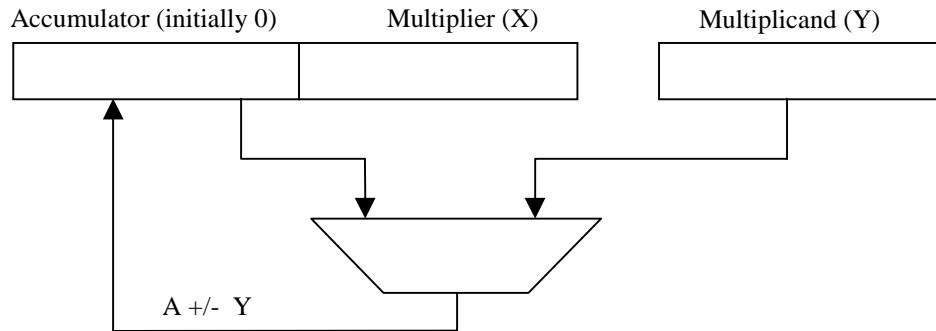
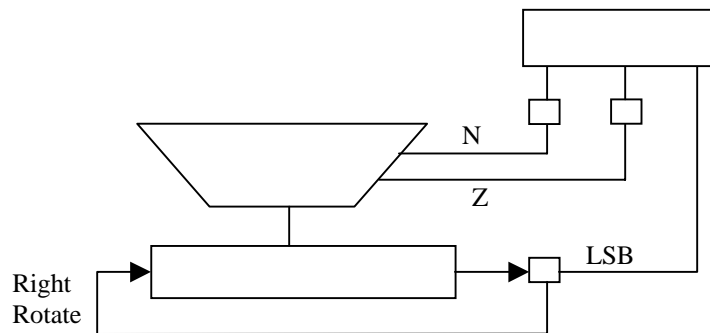


1. Consider adding a 2's complement integer multiply ISA instruction to the IJVM implemented by the MIC-1. The instruction would pop the top two numbers off the stack, multiply them, and store the least significant portion, and the most significant portion of the product back on the stack. An unmodified shift-and-add Booth's algorithm multiply can map on to the MIC-1 registers as follows:



To apply the algorithm, the hardware must handle two features:

1. The LSB of X must be able to be captured as the next recode bit,
  2. The TOS and MDR registers must be able to act as a single 64-bit shift register.
- Both features can be accommodated if we modify the MIC-1 hardware as follows:



We add a Flip-Flop to capture the LSB after a SRA1 shifter operation, and we add a right rotate operation (RROT) as shown, that will be applied when both shifter control bits are high. New logic for the High Bit will now be:

$$\text{High Bit} = (\text{N and JAMN}) \text{ or } (\text{Z and JAMZ}) \text{ or } (\text{not-LSB and JAMZ}) \text{ or } \text{NEXT-ADDR}(8).$$

This gives two new micro-operations: “**RROT**” and “**if (not-LSB) goto n**”.

With the hardware as defined above, write the micro-code for the new MULT instruction, using symbolic addresses for the micro-instructions and assuming sequential execution except where jumps are indicated. i.e we assume a micro-assembler will handle generating binary addresses for us.

2. In the Mic-1 microprogram there are 112 total entries, of which 67 are unique. Compute the % saving in total ROM bits if a nano-programming approach were used, in which the next address is obtained from the micro-store, and the micro-instruction control bits are obtained from a nano-store.