9 bit CPU

## ISA Requirements

Your instruction set architecture shall feature fixed-length instructions (machine code) 9 bits wide and a data path 8 bits wide.

Your ISA specification should describe:

* What operations it supports and what their respective opcodes are.
  + For ideas, see the MIPS, ARM, RISC-V, and/or SPARC instruction lists
* How many instruction formats it supports and what they are
  + **In detail!** How many bits for each field, where they are found in the instruction.
  + Your instruction format description should be detailed enough that someone other than you could write an assembler (a program that creates machine code from assembly code) for it. (Again, refer to ARM or MIPS.)
* Number of registers, and how many general-purpose or specialized.
* All internal **data** paths and storage will be 8 bits wide.
* Addressing modes supported
  + This applies to both memory instructions and branch instructions.
  + How are addresses constructed or calculated? Lookup tables? Sign extension? Direct addressing? Indirect? Immediates?

## Architecture Limitations and Requirements

We shall impose the following constraints on your design, which will make the design a bit simpler:

1. Your core should have separate instruction memory and data memory.
2. You should assume single-ported data memory (a maximum of one read or one write per instruction, not both — Your data memory will have only one address pointer input port, for both input and output). You can write and read in place.
3. Your instruction memory *should* not exceed 2^10 entries; it *must* not exceed 2^12 entries. If you need the larger number of instruction entries, your writeup *must* explain how these extra entries improve some other performance element.
4. Your data memory must not exceed 2^8 entries.
5. You should also assume a register file (or whatever internal storage you support) that can write to only one register per instruction.
   1. The sole exception to this rule is that you may have a multibit ALU condition/flag register (e.g., carry out, or shift out, sign result, zero bit, etc., like ARM's Z, N, C, and V status bits) that can be written at the same time as an 8-bit data register, if you want.
   2. You may read up to two data registers per cycle.
   3. Your register file will have no more than two data output ports and one data input port.
   4. You may use separate pointers for reads and writes, if you wish.
   5. Please restrict register file size to no more than 16 registers.
6. Manual loop unrolling of your code is not allowed – use at least some branch or jump instructions.
7. Your ALU instructions will be a subset of those in ARMsim, or of comparable complexity.
8. You may use lookup tables / decoders, but these are limited to 32 elements each (i.e., pointer width up to 5 bits).
   1. You may not, for example, build a big 512-element, 32-bit LUT to map your 9-bit machine codes into ARM- or MIPS-like wider microcode. (It was amusing the first time a team tried it, but it got old ☺.)

## What must the processor do?

### **Program 1**

**Closest and farthest Hamming pairs** -- Write a program to find the least and greatest Hamming distances among all pairs of values in an array of 32 two-byte half-words. Assume all values are signed 16-bit (“half-word”) integers. The array of integers runs from data memory location 0 to 63. Even-numbered addresses are MSBs, following odd addresses are LSBs, e.g. a concatenation of addresses 0 and 1 forms a 16-bit two’s complement half-word. Write the minimum distance in location 64 and the maximum in 65.

### **Program 2**

**Closest and farthest arithmetic pairs** -- Write a program to find the absolute values of the least and greatest arithmetic difference among all pairs of incoming values from Program 2. Assume again that all values are two’s complement (“signed”) 16-bit integers. The array of integers starts at location 0. Write the absolute value of the minimum difference in locations 66-67 and the maximum in 68-69. Format: mem[66] = MSB of smallest absolute value difference among pairs; mem[67] = LSB. mem[68] = MSB of largest absolute value difference among pairs, mem[69] = LSB.